ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

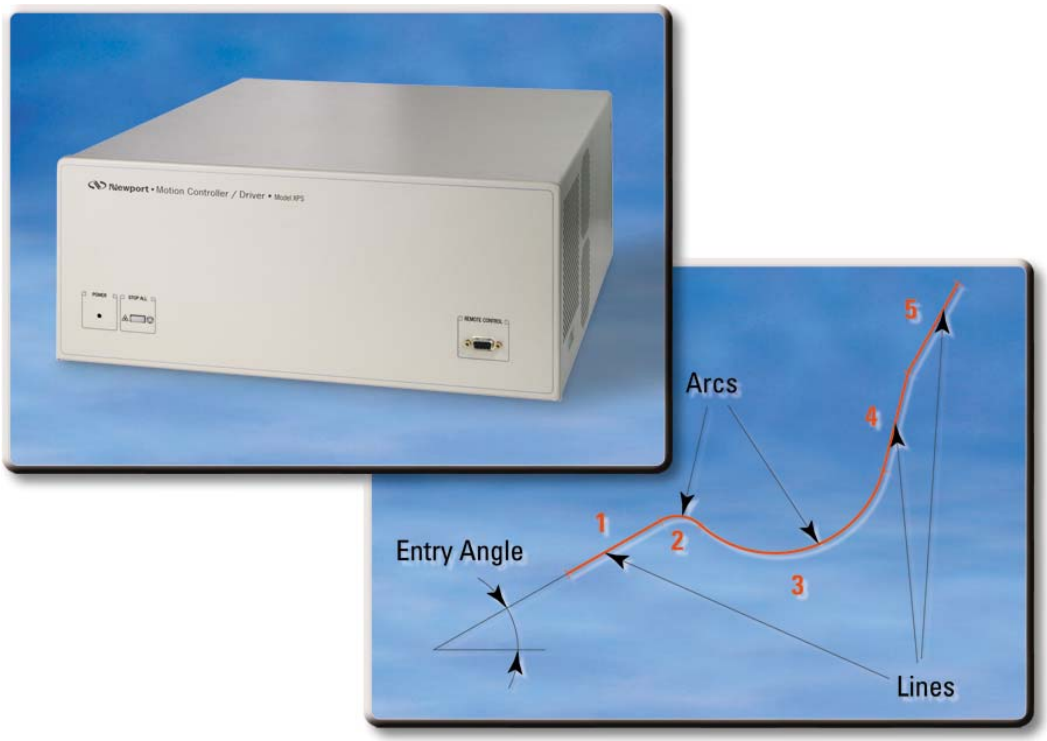
Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Newport XPS-C8* at our website: ***Click HERE***

XPS-Q8

Universal High-Performance Motion Controller/Driver



User's Manual, Software Tools and Tutorial

V1.2.x

Newport®
Experience | Solutions

For Motion, Think Newport™

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table of Contents

Waranty ix
EU Declaration of Conformity x
Preface xi
Confidentiality & Proprietary Rights xi
Sales, Tech Support & Service xi
Service Information xii
Newport Corporation RMA Procedures xii
Packaging xii

User's Manual

1.0 Introduction 1
1.1 Scope of the Manual 1
1.2 Definitions and Symbols 3
1.2.1 General Warning or Caution 3
1.2.2 Electric Shock 3
1.2.3 European Union CE Mark 3
1.2.4 "ON" Symbol 3
1.2.5 "OFF" Symbol 3
1.3 Warnings and Cautions 4
1.4 General Warnings and Cautions 4

2.0 System Overview 6
2.1 Specifications 6
2.2 Drive Options 7
2.3 Compatible Newport Positioners and Drive Power Consumption 8
2.4 XPS Hardware Overview 9
2.5 Front Panel Description 9
2.6 Rear Panel Description 10
2.6.1 Axis Connectors (AXIS 1 – AXIS 8) 10
2.7 Ethernet Configuration 11
2.7.1 Communication Protocols 11
2.7.2 Addressing 12
2.8 Sockets, Multitasking and Multi-user Applications 12
2.9 Programming with TCL 12

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.0 Getting Started 14
3.1 Unpacking and Handling 14
3.2 Inspection for Damage 14
3.3 Packing List 14
3.4 System Setup 14
3.4.1 Installing Driver cards 15
3.4.2 Power ON 15
3.5 Connecting to the XPS 16
3.5.1 Straight through cables (black) 16
3.5.2 Cross-over cables (gray) 16
3.5.3 Direct Connection to the XPS controller 17
3.5.4 Connecting the XPS to a Corporate Network using Static IP Configuration 19
3.5.5 Connecting the XPS to a Corporate Network using Dynamic IP Configuration 21
3.5.6 Recovering a lost IP configuration 22
3.6 Testing your XPS-PC Connection and Communication 24
3.7 Connecting the Stages 25
3.8 Configuring the Controller 26
3.8.1 Auto Configuration 27
3.8.2 Manual Configuration for Newport Positioners 29
3.8.3 Manual Configuration for non Newport stages 33
3.9 System Shut-Down 33

Software Tools

4.0 Software Tools 34
4.1 Software Tools Overview 34
4.2 CONTROLLER CONFIGURATION – Users Management 35
4.3 CONTROLLER CONFIGURATION – IP Management 36
4.4 CONTROLLER CONFIGURATION – General 36
4.5 SYSTEM – Error file display 37
4.6 SYSTEM – Last error file display 37
4.7 SYSTEM – Auto Configuration 38
4.8 SYSTEM – Manual Configuration 38
4.9 SYSTEM – Manual Configuration – Gantries (Secondary Positioners) 42
4.9.1 Home search of gantries 43
4.9.2 Gantries with linear motors 44
4.9.3 Gantries with linear motors and variable force ratio 45
4.10 STAGE – Add from Data Base 47
4.11 STAGE – Modify 48
4.12 FRONT PANEL – Move 50
4.13 FRONT PANEL – Jog 51

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.14 FRONT PANEL – Spindle 51
4.15 FRONT PANEL – I/O View 52
4.16 FRONT PANEL – I/O Set 52
4.17 FRONT PANEL – Positioner Errors 53
4.18 FRONT PANEL – Hardware Status 53
4.19 FRONT PANEL – Driver Status 54
4.20 TERMINAL 54
4.21 TUNING – Auto-Scaling 57
4.22 TUNING – Auto-Tuning 58
4.23 FUNCTIONAL TESTS 61
4.24 FTP (File Transfer Protocol) Connection 61

5.0 Maintenance and Service 63
5.1 Enclosure Cleaning 63
5.2 Obtaining Service 63
5.3 Troubleshooting 63
5.4 Updating the Firmware Version of Your XPS Controller 64

Motion Tutorial

6.0 XPS Architecture 65
6.1 Introduction 65
6.2 State Diagrams 66
6.3 Motion Groups 68
6.3.1 Specific SingleAxis Group Features 69
6.3.2 Specific Spindle Group Features 69
6.3.3 Specific XY Group Features 69
6.3.4 Specific XYZ Group Features 69
6.3.5 Specific MultipleAxes Features 69
6.4 Native Units 69

7.0 Motion 71
7.1 Motion Profiles 71
7.2 Home Search 73
7.3 Referencing State 76
7.3.1 Move on sensor events 77
7.3.2 Moves of Certain Displacements 78
7.3.3 Position Counter Resets 78
7.3.4 State Diagram 79
7.3.5 Example: MechanicalZeroAndIndexHomeSearch 79
7.4 Move 79
7.5 Motion Done 81

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

7.6 JOG 83
7.7 Master Slave 84
7.8 Analog Tracking 85
7.8.1 Analog Position Tracking 86
7.8.2 Analog Velocity Tracking 86

8.0 Trajectories 88
8.1 Line-Arc Trajectories 88
8.1.1 Trajectory Terminology 88
8.1.2 Trajectory Conventions 89
8.1.3 Geometric Conventions 89
8.1.4 Defining Line-Arc Trajectory Elements 89
8.1.5 Define Lines 90
8.1.6 Define Arcs 91
8.1.7 Trajectory File Description 91
8.1.8 Trajectory File Examples 91
8.1.9 Trajectory Verification and Execution 92
8.1.10 Examples of the Use of the Functions 93
8.2 Splines 94
8.2.1 Trajectory Terminology 94
8.2.2 Trajectory Conventions 94
8.2.3 Geometric Conventions 94
8.2.4 Catmull-Rom Interpolating Splines 95
8.2.5 Trajectory Elements Arc Length Calculation 95
8.2.6 Trajectory File Description 96
8.2.7 Trajectory File Example 96
8.2.8 Spline Trajectory Verification and Execution 98
8.2.9 Examples 99
8.3 PVT Trajectories 99
8.3.1 Trajectory Terminology 99
8.3.2 Trajectory Conventions 99
8.3.3 Geometric Conventions 100
8.3.4 PVT Interpolation 100
8.3.5 Influence of the Element Output Velocity to the Trajectory 101
8.3.6 Trajectory File Description 102
8.3.7 Trajectory File Example 103
8.3.8 PVT Trajectory Verification and Execution 104
8.3.9 Examples of the Use of the functions 105

9.0 Emergency Brake and Emergency Stop Cases 106

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10.0 Compensation 109
10.1 Backlash Compensation 110
10.2 Linear Error Correction 111
10.3 Positioner Mapping 111
10.4 XY Mapping 114
10.5 XYZ Mapping 116
10.6 “Yaw” Mapping (PP Firmware Version Only) 122
10.7 “Theta” Encoder and XY Correction 125

11.0 Event Triggers 126
11.1 Events 127
11.2 Actions 135
11.3 Functions 140
11.4 Examples 141

12.0 Data Gathering 145
12.1 Time-Based (Internal) Data Gathering 146
12.2 Event-Based (Internal) Data Gathering 148
12.3 Function-Based (Internal) Data Gathering 151
12.4 Trigger-Based (External) Data Gathering 151

13.0 Output Triggers 153
13.1 Triggers on Line-Arc Trajectories 153
13.2 Triggers on PVT Trajectories 155
13.3 Distance, Time Spaced Pulses or AquadB Position Compare 156
13.3.1 Position compare settings and limits of use 156
13.3.2 Even Distance Spaced Pulses Position Compare 157
13.3.3 Compensated Position Compare 161
13.3.4 Time Spaced Pulses (Time Flasher) 164
13.3.5 AquadB Signals on PCO Connector 166

14.0 Control Loops 169
14.1 XPS Servo Loops 169
14.1.1 Servo structure and Basics 169
14.1.2 XPS PIDFF Architecture 171
14.2 Filtering and Limitation 175
14.3 Feed Forward Loops and Servo Tuning 175
14.3.1 Corrector = PIDFFVelocity 175
14.3.2 Corrector = PIDFFAcceleration 177
14.3.3 Corrector = PIDDual FFVoltage 180
14.3.4 Corrector = PIPosition 181

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

15.0 Analog Encoder Calibration 183

16.0 Excitation Signal 188

16.1 Introduction 188
16.2 How to Use the Excitation-Signal Function 188
16.3 Group State Diagram 189
16.4 Function Description 189

17.0 Pre-Corrector Excitation Signal 190

17.1 Description 190
17.2 Pre-corrector excitation signal wave forms 190
17.3 Technical Implementation 192
17.3.1 Use case 192
17.3.2 Implementation 193
17.3.3 Group capsule state diagram modification 194

18.0 Introduction to XPS Programming 195

18.1 TCL Generator 196
18.2 LabVIEW 197
18.3 DLL Drivers 198
18.4 Running Processes in Parallel 199

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendices

19.0 Appendix A: Hardware 202
19.1 Controller 202
19.2 Rear Panel Connectors 203
19.3 Environmental Requirements 203

20.0 Appendix B: General I/O Description 204
20.1 Digital I/Os (All GPIO, Inhibit and Trigger In, and PCO Connectors) 204
20.1.1 Digital Inputs 204
20.1.2 Digital Outputs 205
20.2 Digital Encoder Inputs (Driver Boards & DRV00) 205
20.3 Digital Servitudes (Driver Boards, DRV00 & Analog Encoders Connectors) 205
20.4 Analog Encoder Inputs (Analog Encoder Connectors) 205
20.5 Analog I/O (GPIO2 Connector) 206
20.5.1 Analog Inputs 206
20.5.2 Analog Outputs 206

21.0 Appendix C: Power Inhibit Connector 207

22.0 Appendix D: GPIO Connectors 208
22.1 GPIO1 Connector 208
22.2 GPIO2 Connector 208
22.3 GPIO3 Connector 209
22.4 GPIO4 Connector 209

23.0 Appendix E: PCO Connector 210

24.0 Appendix F: Motor Driver Cards 211
24.1 DC and Stepper Motor Driver XPS-DRV01 211
24.2 Three phase AC brushless driver XPS-DRV02 212
24.3 DC Motor Driver XPS-DRV03 213
24.4 Pass-Through Board Connector (25-Pin D-Sub) XPS-DRV00 213

25.0 Appendix G: Analog Encoder Connector 214

26.0 Appendix H: Trigger IN Connector 215

Service Form 217

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Warranty

Newport Corporation warrants that this product will be free from defects in material and workmanship and will comply with Newport's published specifications at the time of sale for a period of one year from date of shipment. If found to be defective during the warranty period, the product will either be repaired or replaced at Newport's option.

To exercise this warranty, write or call your local Newport office or representative, or contact Newport headquarters in Irvine, California. You will be given prompt assistance and return instructions. Send the product, freight prepaid, to the indicated service facility. Repairs will be made and the instrument returned freight prepaid. Repaired products are warranted for the remainder of the original warranty period or 90 days, whichever comes first.

Limitation of Warranty

The above warranties do not apply to products which have been repaired or modified without Newport's written approval, or products subjected to unusual physical, thermal or electrical stress, improper installation, misuse, abuse, accident or negligence in use, storage, transportation or handling. This warranty also does not apply to fuses, batteries, or damage from battery leakage.

THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR USE. NEWPORT CORPORATION SHALL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM THE PURCHASE OR USE OF ITS PRODUCTS.

Copyright 2012 by Newport Corporation, Irvine, CA. All rights reserved. No part of this manual may be reproduced or copied without the prior written approval of Newport Corporation. This manual is provided for information only, and product specifications are subject to change without notice. Any change will be reflected in future printings.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

EU Declaration of Conformity

XPS

Newport
Experience | Solutions

Year CE mark affixed: 2011

EC Declaration of Conformity

The manufacturer:

MICRO-CONTROLE Spectra-Physics,
1 rue Jules Guesde ZI. Bois de l'Epine - BP189
F-91006 Evry FRANCE

Hereby declares that the product:

Description: "XPS"
Function: Universal High-Performance Motion Controller/Driver
Type of equipment: Electrical equipment for measurement, control and laboratory use

– complies with all the relevant provisions of the Directive 2004/108/EC relating to electro-magnetic compatibility (EMC).
– complies with all the relevant provisions of the Directive 2006/95/EC relating to electrical equipment designed for use within certain voltage limits (Low Voltage)

– was designed and built in accordance with the following harmonised standards:

NF EN 61326-1:2006 « Electrical equipment for measurement, control and laboratory use – EMC requirements – Part 1: General requirements »

NF EN 55011:2007 Class A

NF EN 61000-3-2:2006 +A1:2009 + A2:2009 « Electromagnetic compatibility (EMC) – Part 3-2: Limits - Limits for harmonic current emissions »

NF EN 61010-1:2001 « Safety requirements for electrical equipment for measurement, control and laboratory use – Part 1: General requirements »

– was designed and built in accordance with the following other standards:

NF EN 61000-4-2
NF EN 61000-4-3
NF EN 61000-4-4
NF EN 61000-4-5
NF EN 61000-4-6
NF EN 61000-4-11

Date : 10/06/2011

Dominique DEVIDAL
Quality Director

MICRO-CONTROLE Spectra-Physics
Zone Industrielle
F-45340 Beaune La Rolande, France

DC2-EN rev:A

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Preface

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.

Sales, Tech Support & Service

North America & Asia
Newport Corporation
1791 Deere Ave.
Irvine, CA 92606, USA

Sales
Tel.: (877) 835-9620
e-mail: sales@newport.com

Technical Support
Tel.: (800) 222-6440
e-mail: tech@newport.com

Service, RMAs & Returns
Tel.: (800) 222-6440
e-mail: rma.service@newport.com

Europe
MICRO-CONTROLE Spectra-Physics S.A.S
9, rue du Bois Sauvage
91055 Évry CEDEX
France

Sales France
Tel.: +33 (0)1.60.91.68.68
e-mail: france@newport-fr.com

Sales Germany
Tel.: +49 (0) 61 51 / 708 – 0
e-mail: germany@newport.com

Sales UK
Tel.: +44 (0)1635.521757
e-mail: uk@newport.com

Technical Support
e-mail: tech_europe@newport.com

Service & Returns
Tel.: +33 (0)2.38.40.51.55

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Service Information

The user should not attempt any maintenance or service of the XPS Series Controller/Driver system beyond the procedures outlined in this manual. Any problem that cannot be resolved should be referred to Newport Corporation. When calling Newport regarding a problem, please provide the Tech Support representative with the following information:

- Your contact information.
- System serial number or original order number.
- Description of problem.
- Environment in which the system is used.
- State of the system before the problem.
- Frequency and repeatability of problem.
- Can the product continue to operate with this problem?
- Can you identify anything that may have caused the problem?

Newport Corporation RMA Procedures

Any XPS Series Controller/Driver being returned to Newport must be assigned an RMA number by Newport. Assignment of the RMA requires the item's serial number.

Packaging

XPS Series Controller/Driver being returned under an RMA must be securely packaged for shipment. If possible, re-use the original packaging.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



User's Manual

1.0 Introduction

1.1 Scope of the Manual

The XPS is an extremely high-performance, easy to use, integrated motion controller/driver offering high-speed communication through 10/100 Base-T Ethernet, outstanding trajectory accuracy and powerful programming functionality. It combines user-friendly web interfaces with advanced trajectory and synchronization features to precisely control from the most basic to the most complex motion sequences. Multiple digital and analog I/O's, triggers and supplemental encoder inputs provide users with additional data acquisition, synchronization and control features that can improve the most demanding motion applications.

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

The **XPS Quick Start** and **XPS User's Manual** are delivered as paper copies with the controller.

The Programmer's, TCL, Software Drivers and Stage Configuration manuals are PDF files accessible from the XPS web site.

DLLs and corresponding sources are available from the controller disk in the folder Public/Drivers/DLL. DLLs can also be downloaded through the FTP.

LabVIEW VIs with examples are also available from the controller disk in the folder Public/Drivers/LabView. They can also be downloaded through FTP.

To connect through FTP, please see chapter 5: "FTP connection".

The first part of this manual serves as an introduction and also as a reference. It includes:

1. Introduction
2. System Overview
3. Getting Started Guide

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The second part provides a detailed description of all software tools of the XPS controller. It also includes an introduction to FTP connections and some general guidelines for troubleshooting, maintenance and service:

4. Software Tools
5. FTP connection
6. Maintenance and Service

The third part provides an exhaustive description of the XPS architecture, its features and capabilities. Complementing the programmer's guide, this part is educational and is organized by features starting with the basics and getting to the more advanced features. It provides a complete list of descriptions of different features including:

7. XPS Architecture
8. Motion
9. Trajectories
10. Compensation
11. Event Triggers
12. Data Gathering
13. Triggers
14. Control Loops
15. Analog Encoder Calibration
16. Introduction to XPS programming

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.2 Definitions and Symbols

The following terms and symbols are used in this documentation and also appear on the XPS Series Controller/Driver where safety-related issues occur.

1.2.1 General Warning or Caution



Figure 1: General Warning or Caution Symbol.

The Exclamation Symbol in Figure 1 may appear in Warning and Caution tables in this document. This symbol designates an area where personal injury or damage to the equipment is possible.

1.2.2 Electric Shock



Figure 2: Electrical Shock Symbol.

The Electrical Shock Symbol in Figure 2 may appear on labels affixed to the XPS Series Controller/Driver. This symbol indicates a hazard arising from dangerous voltages. Any mishandling could result in damage to the equipment, personal injury, or even death.

1.2.3 European Union CE Mark



Figure 3: CE Mark.

The presence of the CE Mark on Newport Corporation equipment means that it has been designed, tested and certified to comply with all current and applicable European Union (CE) regulations and recommendations.

1.2.4 "ON" Symbol



Figure 4: "ON" Symbol.

The "ON" Symbol in Figure 4 appears on the power switch of the XPS Series Controller/Driver. This symbol represents the "Power On" condition.

1.2.5 "OFF" Symbol



Figure 5: "OFF" Symbol.

The "Off" Symbol in Figure 5 appears on the power switch of the XPS Series Controller/Driver. This symbol represents the "Power Off" condition.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.3 Warnings and Cautions

The following are definitions of the Warnings, Cautions and Notes that may be used in this manual to call attention to important information regarding personal safety, safety and preservation of the equipment, or important tips.



WARNING

Situation has the potential to cause bodily harm or death.



CAUTION

Situation has the potential to cause damage to property or equipment.

NOTE

Additional information the user or operator should consider.

1.4 General Warnings and Cautions

The following general safety precautions must be observed during all phases of operation of this equipment.

Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and the intended use of the equipment.

- Heed all warnings on the unit and in the operating instructions.
- To prevent damage to the equipment, read the instructions in this manual for the selection of the proper input voltage.
- Only plug the Controller/Driver unit into a grounded power outlet.
- Ensure that the equipment is properly grounded to earth ground through the grounding lead of the AC power connector.
- Route power cords and cables where they are not likely to be damaged.
- The system must be installed in such a way that the power switch and the power connector remain accessible to the user.
- Disconnect or do not plug-in the AC power cord under the following conditions:
  - If the AC power cord or any other attached cables are frayed or damaged.
  - If the power plug or receptacle is damaged.
  - If the unit is exposed to rain or excessive moisture, or liquids are spilled on it.
  - If the unit has been dropped or the case is damaged.
  - If the user suspects service or repair is required.
- Keep air vents free of dirt and dust and obstructions.
- Keep liquids away from unit.
- Do not expose equipment to excessive moisture (>85% humidity).
- Do not operate this equipment in an explosive atmosphere.
- Disconnect power before cleaning the Controller/Driver unit. Do not use liquid or aerosol cleaners.
- Do not open the XPS Controller/Driver stand alone motion controller. There are no user-serviceable parts inside the XPS Controller/Driver.
- Return equipment to Newport Corporation for service and repair.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Dangerous voltages associated with the 100–240 VAC power supply are present inside Controller/Driver unit. To avoid injury, do not touch exposed connections or components while power is on.
- Follow precautions for static-sensitive devices when handling electronic circuits.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.0 System Overview

2.1 Specifications

Number of Axes	• 1 to 8 axes of stepper, DC brush, DC brushless motors or piezo-electric stacks using internal drives • Other motion devices using external third-party drives
Communication Interfaces	• Internet protocol TCP/IP • One Ethernet 10/100 Base-T (RJ45 connector) with fixed IP address for local communication • One Ethernet 10/100 Base-T (RJ45 connector) for networking, dynamic addressing with DHCP and DNS • Typically 0.3 ms from sending a tell position command to receiving the answer • Optional XPS-RC remote control
Firmware Features	• Powerful and intuitive, object oriented command language • Native user defined units (no need to program in encoder counts) • Real time execution of custom tasks using TCL scripts • Multi-user capability • Concept of sockets for parallel processes • Distance spaced trigger output pulses, max. 2.5 MHz rate, programmable filter • Time spaced trigger output pulses, 0.02 Hz to 2.5 MHz rate, 50 ns accuracy • Trigger output on trajectories with 100 µs resolution • Data gathering at up to 10 kHz rate, up to 1,000,000 data entries • User-defined "actions at events" monitored by the controller autonomously at a rate of 10 kHz • User-definable system referencing with hardware position latch of reference signal transition and "set current position to value" capability • Axis position or speed controlled by analog input • Axis position, speed or acceleration copied to analog output • Trajectory precheck function replying with travel requirement and max. possible speed • Auto-tuning and auto-scaling
Motion	• Jogging mode including on-the fly changes of speed and acceleration • Synchronized point-to-point • Spindle motion (continuous motion with periodic position reset) • Gantry mode including XY gantries with variable load ratio • Line-arc mode (linear and circular interpolation incl. continuous path contouring) • Splines (Catmull-Rom type) • PVT (complex trajectory based on position, velocity and time coordinates) • Analog tracking (using analog input as position or velocity command) • Master-slave including single master-multiple slaves and custom gear ratio
Compensation	• Linear error, Backlash, positioner error mapping • XY and XYZ error mapping • All corrections are taken into account on the servo loop
Servo Rate	• 10 kHz
Control Loop	• Open loop, PI position, PIDFF velocity, PIDFF acceleration, PIDDualFF voltage • Variable PID's (PID values depending on distance to target position) • Deadband threshold; Integration limit and integration time • Derivative cut-off filter; 2 user-defined notch filters

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

I/O	• 30 TTL inputs and 30 TTL outputs (open-collector) • 4 synch. analog inputs ±10 V, 14 Bit • 4 synch. uncommitted analog outputs, 16 Bit • Watchdog timer and remote interlock
Trigger In	• Hardware latch of all positions and all analog I/O's; 10 kHz max. frequency • <50 ns latency on positions • <100 µs time jitter on analog I/O's
Trigger Out	• One high-speed position compare output per axes that can be either configured for position synchronized pulses or for time synchronized pulses : <50 ns accuracy/latency, 2.5 MHz max. rate
Dedicated Inputs Per Axis	• RS-422 differential inputs for A, B and I, Max. 25 MHz, over-velocity and quadrature error detection • 1 Vpp analog encoder input up to x32768 interpolation used for servo; amplitude, phase and offset correction; additional 2nd hardware interpolator used for synchronization; up to x200 interpolation • Forward and reverse limit, home, error input
Dedicated Outputs Per Axis (when using external drives)	2 channel 16-bit, ±10 V D/A Drive enable, error output
Drive Capability	• Analog voltage, analog velocity, and analog acceleration (used with XPS-DRV01 and XPS-DRV03 for DC brush motor control). • Analog position (used with XPS-DRV01 for stepper motor control or with the XPS-DRVP1 for piezo control) • Analog position (used with external drives for example 3rd party motors) • Analog acceleration, sine acceleration and dual sine acceleration (used with XPS-DRV02 for brushless motors control) • Step and direction and +/- pulse mode for stepper motors (requires XPS-DRV00P and external stepper motor driver) • 500 W @ 230 VAC and 425 W @ 115 VAC total available power
Dimensions (W x D x H)	• 19" – 4U, L: 508 mm
Weight	• 15 kg max

2.2 Drive Options

The XPS controller is capable of driving up to 8 axes of most Newport positioners using driver cards that slide through the back of the chassis. These factory-tested drives are powered by an internal 500 W power supply, which is independent of the controller power supply. When used with Newport ESP stages, the configuration of the driver cards is easy using the auto-configuration utility software. Advanced users can also manually develop their own configuration files optimized for specific applications.

The XPS-DRV01 is a software configurable PWM amplifier that is compatible with most of Newport's and other companies' DC brush and stepper motor positioners.

The XPS-DRV01 motor driver supplies a maximum current of 3 Amps and 48 Volts. It has the capability to drive bipolar stepper motors in microstep mode (sine/cosine commutation) and DC brush motors in velocity mode, for motors with tachometer, or voltage mode, for motors without tachometer. Programmable gains and a programmable PWM switching frequency up to 300 kHz allow a very fine adjustment of the driver to the motor. For added safety, a programmable over-current protection setting is also available.

The XPS-DRV02 is a software configurable PWM amplifier for 3-phase brushless motors. It has been optimized for performance with XM, ILS-LM, IMS-LM linear motor stages and RGV direct drive rotation stages. The XPS-DRV02 supplies a 100 kHz PWM output with a maximum output current of 5 A per phase and 44 Vpp. The

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

XPS-DRV02 requires 1 Vpp analog encoder input signals used also for motor commutation. Motor initialization is done by a special routine measures the magnetic position without the need for Hall or other sensors.

The XPS-DRV03 is a fully digital, programmable PWM-amplifier that has been optimized for use with high-performance DC motors. The high switching frequency of 100 kHz and appropriate filter technologies minimize noise to enable ultra-precision positioning in the nm-range. The XPS-DRV03 supplies a maximum current of 5 Amps and 48 Volts. It is capable of driving DC motors in velocity mode (for motors with tachometer), in voltage mode (for motors without tachometer), and in current mode (for torque motors). All parameters are programmable in physical units (for instance the bandwidth of the velocity loop). Furthermore, the XPS-DRV03 features individual limits for the rms current and the peak current.

The XPS-DRVP1 is a programmable driver card for Newport's NanoPositioning line of piezoelectric stack stages. This driver card has a range of -10 to 150 VDC with 30 mA continuous. The drive features a 4 kHz update rate and resolution of 16 bits ADC and DAC. It also accepts strain gage position feedback.

The XPS-DRV00 and XPS-DRV00P pass-through module can be used to pass control signals to other external third-party amplifiers (drivers). By setting the controller's dual DAC output to either analog position, analog stepper position, analog velocity, analog voltage or analog acceleration (including sine commutation), the XPS is capable of controlling almost any motion device including 3rd party brushless motors and voice coils.

In addition to conventional digital AquadB feedback encoder interface, the XPS controller also features a high-performance analog encoder input (1 Vpp Heidenhain standard) on each axis. An ultra-high resolution, very low noise, encoder signal interpolator converts the sine-wave input to an exact position value with a signal subdivision up to 32,768-fold. For example, when used with a scale with 4 µm signal period the resolution can be as fine as 0.122 nm. This interpolator can be used for accurate position feedback on the servo corrector of the system. An additional hardware interpolator with 40 MHz clock frequency and programmable signal subdivision up to 200-fold is used for synchronization purposes. This fast interpolator latches the position directly with less than 50 ns latency and provides a much higher level of precision for synchronization than alternative time based systems. Unlike most high-resolution multiplication devices, the XPS interpolators do not compromise positioning speed. With a maximum input frequency ranging from 180 kHz to 400 kHz (depending on the interpolation factor), the maximum speed of a stage with a 20 µm signal period scale can be up to 3.6 m/s.

2.3 Compatible Newport Positioners and Drive Power Consumption

The list of all compatible Newport positioners and the corresponding drive module needed is available from the Newport catalog or at www.newport.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.4 XPS Hardware Overview

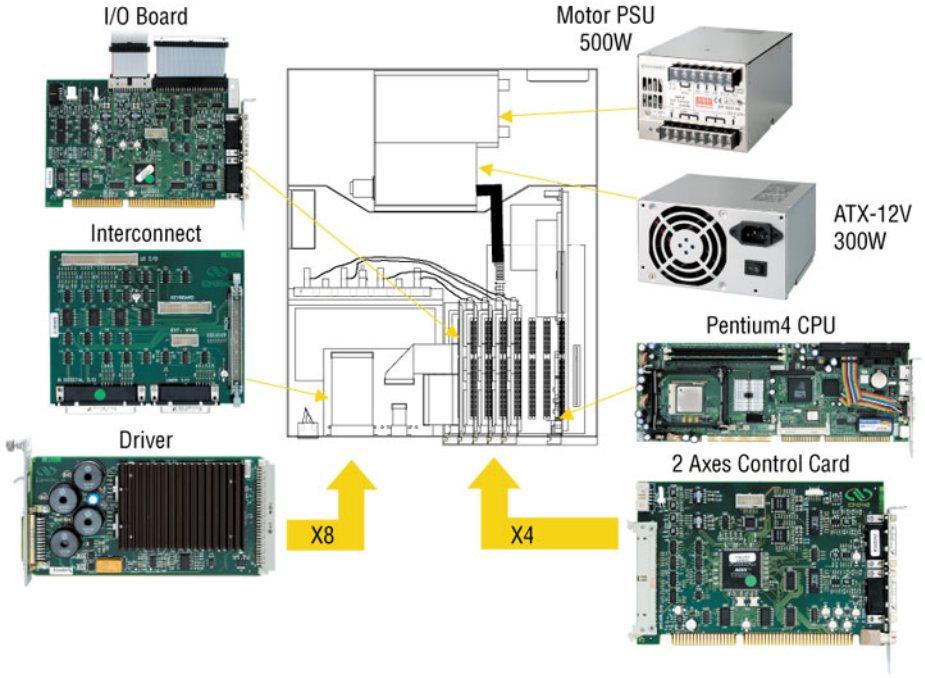


Figure 6: XPS Hardware Overview.

2.5 Front Panel Description



Figure 7: Front Panel of XPS Controller/Driver.

The XPS-RC Remote Control plugs into the front panel of the XPS controller to enable computer-independent motion and basic system diagnostics. For more information, refer to the XPS data sheet and the XPS-RC manual.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.6 Rear Panel Description

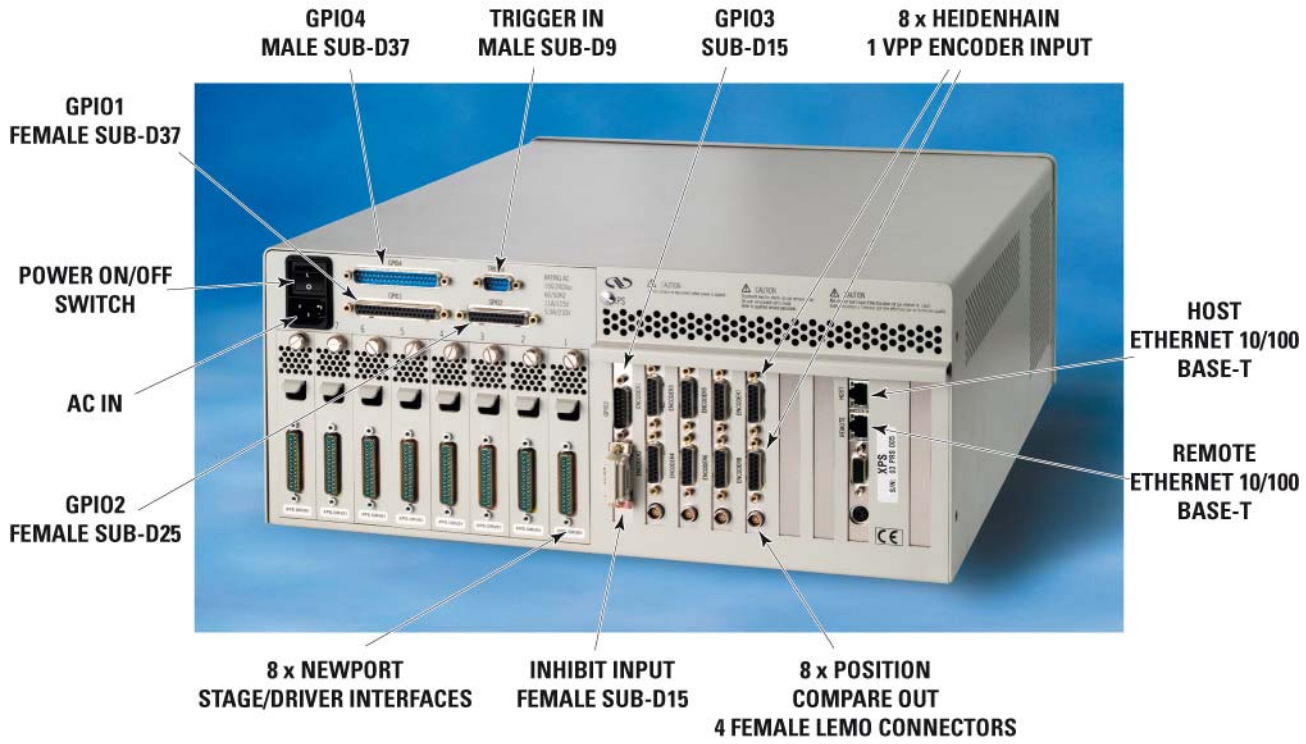


Figure 8: Rear Panel of XPS Controller/Driver.

NOTE

The Main Power ON/OFF Switch is located above the inlet for the power cord. The switch and the inlet must be accessible to the user.

2.6.1 Axis Connectors (AXIS 1 – AXIS 8)

Each installed axis driver card features a connector to attach a cable (supplied with every Newport stage) between the controller and a motion device.

CAUTION



Carefully read the labels on the driver cards and make sure the specifications (motor type, voltage, current, etc.) match those of the motion devices you intend to connect. Severe damage could occur if a stage is connected to the wrong driver card.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Figure 9: Axis Driver Card.

Please see the next section for installation instructions.

NOTE

Power Input: 100–240 V, 50–60 Hz, 11 A–5.5 A.

2.7 Ethernet Configuration



Figure 10: Ethernet Configuration.

2.7.1 Communication Protocols

The Ethernet connection provides a local area network through which information is transferred in units known as packets. Communication protocols are necessary to dictate how these packets are sent and received. The XPS Controller/Driver supports the industry standard protocol TCP/IP.

TCP/IP is a "connection" protocol and in this protocol, the master must be connected to the slave in order to begin communication. Each packet sent is acknowledged when received. If no acknowledgment is received, the information is assumed lost and is resent.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.7.2 Addressing

There are two levels of addresses that define Ethernet devices. The first is the MAC address. This is a unique and permanent 6 byte number. No other device will have the same MAC address. The second level of addressing is the IP address. This is a 32-bit (or 4 byte) number. The IP address is constrained by each local network and must be assigned locally. Assigning an IP address to the controller can be done in a number of ways (see section 3.5: "Connecting to the XPS").

2.8 Sockets, Multitasking and Multi-user Applications

Based on the TCP/IP Internet communication protocol, the XPS controller has a high number of virtual communication ports, known as sockets. To establish communication, the user must first request a socket ID from the XPS controller server (listening at a defined IP number and port number). When sending a function to a socket, the controller will always reply with a completion or error message to the socket that has requested the action.

The concept and application of sockets has many advantages. First, users can split their application into different segments that run independently on different threads or even on different computers. To illustrate this, see below:

```
SocketID1=OpenSocket (...)                    SocketID2=OpenSocket (...)
…                                             …
…                                             …
For i = 1 to nbpos                            Zerror=ReadAFSensor
   Goal=Position (i)                          error=GroupMoveRelative (SocketID2, Z, Zerror)
   error=GroupMoveAbsolute (SocketID1, XY, goal)  …
   if error=OK than TakePicture               …
   …
Next i
…
```

In this example, a thread on socket 1 commands an XY stage to move to certain positions to take pictures while another thread on socket 2 independent of socket 1, concurrently manages an auto-focusing system. The second task could even be run on a different PC than the first task yet be simultaneously executed within the XPS. Alternatively, if the auto-focusing system is providing an analog feedback, this task could have been also implemented as a TCL script within the XPS (see the next topic).

Second, the concept of sockets has another practical advantage for many laboratory users since the use of threads allows them to share the same controller for different applications at the same time. With the XPS, it is possible that one group uses one axis of the XPS controller for an optical delay line, while another group simultaneously uses other axes for a totally different application. Both applications could run completely independent from different workstations without any delays or cross-talk.

The XPS controller uses TCP/IP blocking sockets, which means that the commands to the same socket are "blocked" until the XPS returns feedback about the completion of the currently executed command (either '0' if the command has been completed successfully, or an error code in case of an error). If customers want to run several processes in parallel, users should open as many 84 parallel sockets. Please refer to section 18.4: "Running Processes in Parallel" for further information about sockets and parallel processing.

2.9 Programming with TCL

TCL documentation is in a PDF file accessible from the XPS controller web site.

TCL stands for Tool Command Language and is an open-source string based command language. With only a few fundamental constructs and relatively little syntax, it is very easy to learn, yet it can be as powerful and functional as traditional C language. TCL includes many different math expressions, control structures (if, for, foreach, switch, etc.), events, lists, arrays, time and date manipulation, subroutines, string manipulation, file management and much more. TCL is used worldwide with a user base approaching

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

one million users. It is quickly becoming a standard and critical component in thousands of corporations. Consequently TCL is field proven, very well documented and has many tutorials, applications, tools and books publicly available (www.tcl.tk).

XPS users can use TCL to write complete application code and the XPS allows them to include any function in a TCL script. When developed, the TCL script can be executed in real time in the background of the motion controller processor and does not impact any processing requirements for servo updates or communication. The QNX hardware real time multiprocessing operating system used on the XPS controller assures precise management of the multiple processes with the highest reliability. Multiple TCL programs run in a time-sharing mode with the same priority and will get interrupted only by the servo, or communication tasks or when the maximum available time of 20 ms for each TCL program is over.

The advantage of executing application code within the controller over host run code is faster execution and better synchronization, in many cases without any time taken from the communication link. The complete communication link can be reserved for time critical process interaction from or to the process or host controller.

NOTE

It is important to note that the XPS gives communication requests priority over TCL script execution. When using TCL scripts for machine security or other time critical tasks, it is therefore important to limit the frequency of continuous communication requests from a host computer, which includes the XPS website, and to verify the execution speed of repetitive TCL scripts.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.0 Getting Started

3.1 Unpacking and Handling

It is recommended that the XPS Controller/Driver be unpacked in your lab or work site rather than at the receiving dock. Unpack the system carefully; small parts and cables are included with the equipment. Inspect the box carefully for loose parts before disposing of the packaging. You are urged to save the packaging material in case you need to ship your equipment.

3.2 Inspection for Damage

XPS Controller/Driver has been carefully packaged at the factory to minimize the possibility of damage during shipping. Inspect the box for external signs of damage or mishandling. Inspect the contents for damage. If there is visible damage to the equipment upon receipt, inform the shipping company and Newport Corporation immediately.

WARNING



Do not attempt to operate this equipment if there is evidence of shipping damage or you suspect the unit is damaged. Damaged equipment may present additional personnel hazard. Contact Newport technical support for advice before attempting to plug in and operate damaged equipment.

3.3 Packing List

Included with each XPS controller are the following items:

- User's Manual and Motion Tutorial.
- XPS controller.
- Cross-over cable, gray, 3 meters.
- Straight-through cable, black, 5 meters.
- Power cord.
- Rack mount ears and handles.

If there are missing hardware or have questions about the hardware that were received, please contact Newport.



CAUTION

Before operating the XPS controller, please read chapter 1.0 very carefully.

3.4 System Setup

This section guides the user through the proper set-up of the motion control system. If not already done, carefully unpack and visually inspect the controller and stages for any damage. Place all components on a flat and clean surface.



CAUTION

No cables should be connected to the controller at this point!

First, the controller must be configured properly before stages can be connected.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.4.1 Installing Driver cards

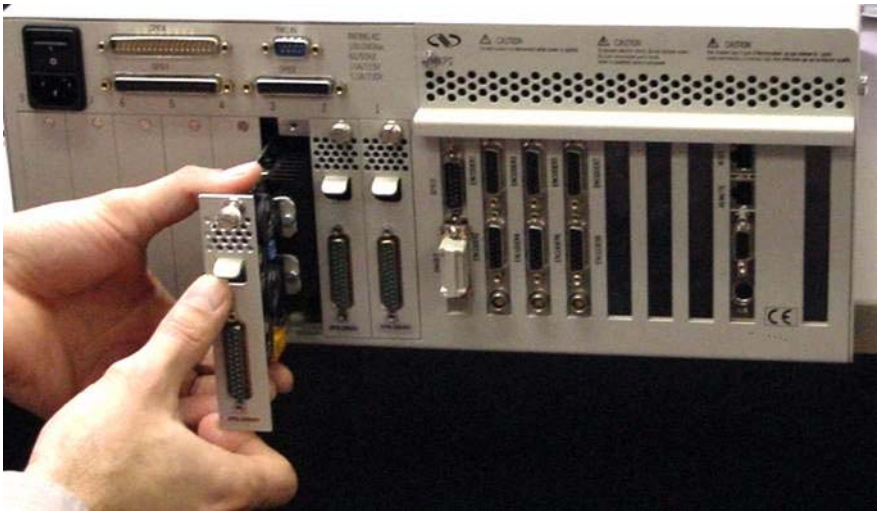


Figure 11: Installing Driver cards.

Due to the high power of the XPS controller (300 W for the CPU and 500 W for the drives), ventilation is very important.

To ensure a good level of heat dissipation, the following rules must be followed:

1. It is strictly forbidden to use the XPS controller without the cover properly mounted on the chassis.
2. Driver boards must be inserted from right (driver 1) to left (driver 8) when looking at the rear of the controller.
3. If less than eight are used, the remaining slots must be disabled with the appropriate slot covers that were delivered with the controller.
4. The surrounding ventilation holes at the sides and back of the XPS rack must be free from obstructions that prevent the free flow of air.

3.4.2 Power ON

- Plug the AC line cord supplied with the XPS into the AC power receptacle on the rear panel.
- Plug the AC line cord into the AC wall-outlet. Turn the Main Power Switch to ON (located on the Rear Panel).
- The system must be installed in such a way that power switch and power connector are accessible by the user.
- After the main power is switched on, the LED on the front panel of the XPS will turn green.
- There is an initial beep after power on and a second beep when the controller has finished booting. If the controller boots properly, the second beep is happy-sounding, otherwise the sad-sounding beep is emitted. The time between the first and the second beeps can be 12–18 seconds.
- There is also a STOP ALL button on the front panel that is used to shut down all the motors.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5 Connecting to the XPS

The Newport's XPS Controller/Driver is a multi-axis motion controller system that is based on a high performance 10/100 Base-T Ethernet connection using a CAT5 cable.

The controller can be connected in 2 different ways:

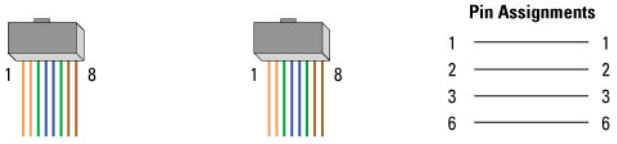
1. Direct connection-PC to XPS through a cross over cable (gray cable supplied).
2. Corporate Network connection – requires input from a Network Administrator (black).

Two cables are provided with the motion controller:

- Cross-over cable – used when connecting the XPS directly to a PC.
- Straight Ethernet cable – used when connecting the XPS through an intranet.

3.5.1 Straight through cables (black)

Standard Ethernet straight through cables are required when connecting the device to a standard network hub or switch.



1 – Orange/White
2 – Orange
3 – Green/White
4 – Blue (not used)
5 – Blue/White (not used)
6 – Green
7 – Brown/White (not used)
8 – Brown (not used)

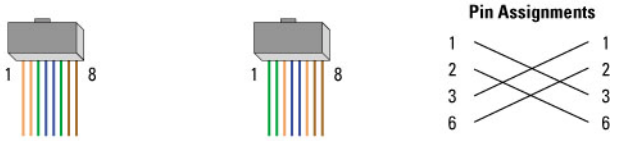
Figure 12: Straight through cables.

3.5.2 Cross-over cables (gray)

Standard Ethernet cross over cables are required when connecting the device directly to the Ethernet port of a PC.

NOTE

Cross over cables are typically labeled (cross over or XO) at one or both ends.



1 – Orange/White
2 – Orange
3 – Green/White
4 – Blue (not used)
5 – Blue/White (not used)
6 – Green
7 – Brown/White (not used)
8 – Brown (not used)

1 – Green/White
2 – Green
3 – Orange/White
4 – Blue (not used)
5 – Blue/White (not used)
6 – Orange
7 – Brown/White (not used)
8 – Brown (not used)

Figure 13: Ethernet Cross Over Cables.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5.3 Direct Connection to the XPS controller

For a direct connection between a PC and the XPS controller you need to use the cross-over cable and the HOST connector at the back of the XPS.

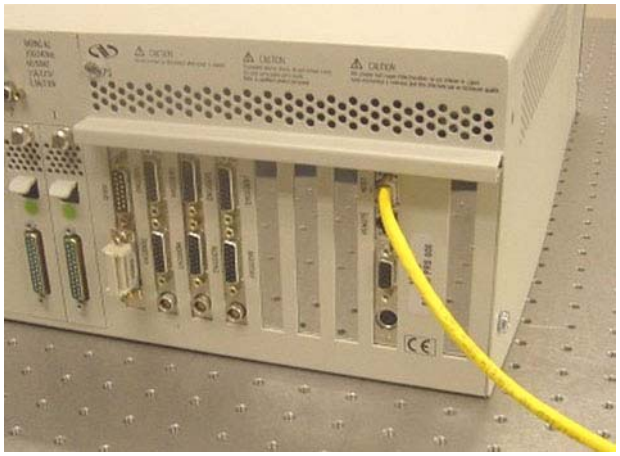
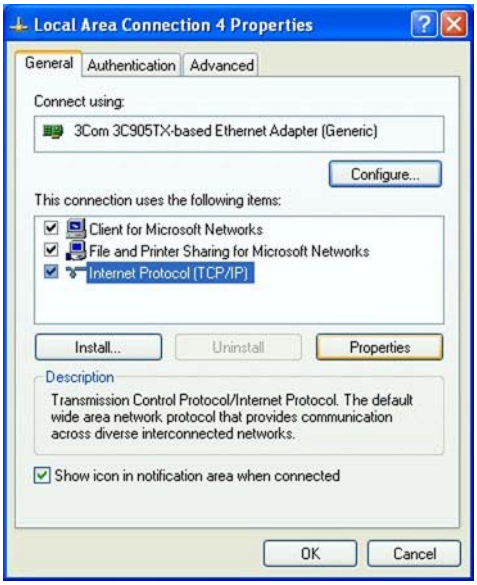


Figure 14: Direct Connection to the XPS using cross-over cable.

First, the IP address on the PC's Ethernet card has to be set to match the default factory XPS's IP address (192.168.0.254). Following is the procedure to set the Ethernet card address.

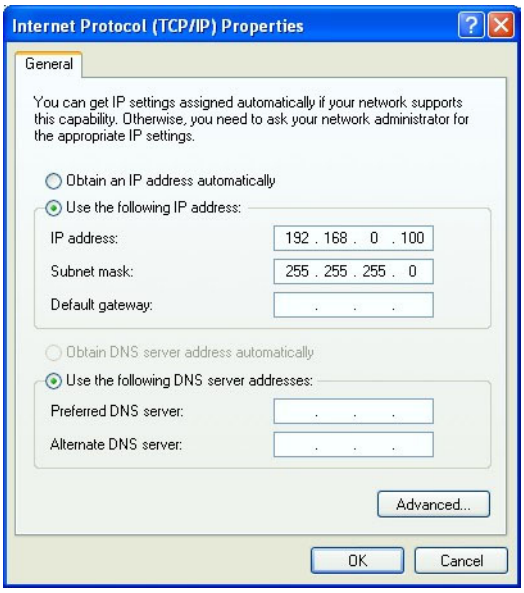
This procedure is for the Windows XP operating system (almost similar process for Window 7):

1. Start Button > Control Panel > Network Connections (Network and Sharing Center => Change adapter settings).
2. Right Click on Local Area Connection Icon and select Properties.



3. Highlight Internet Protocol (TCP/IP, TCP/IP4) and click on Properties.
4. Type the following IP address and Subnet Mask as shown in the next image.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



5. Click "OK".

NOTE

The Last number of the IP address must be set to any number between 2 to 253: 100 for example.

NOTE

When configuring the controller to be on the network, the settings for the PC's Ethernet card will have to be set back to default under "Obtain an IP address automatically".

Once the Ethernet card address is set, you are ready to connect to the XPS controller. Following is the procedure for connecting to the controller:

6. Open Internet Browser and connect to http://192.168.0.254

Login:

Name: **Administrator**

Password: **Administrator** (Please see the picture below).

Rights: **Administrator**

NOTE

Please note that the login text is case sensitive.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Once logged in, the XPS has established a direct connection to the local computer.

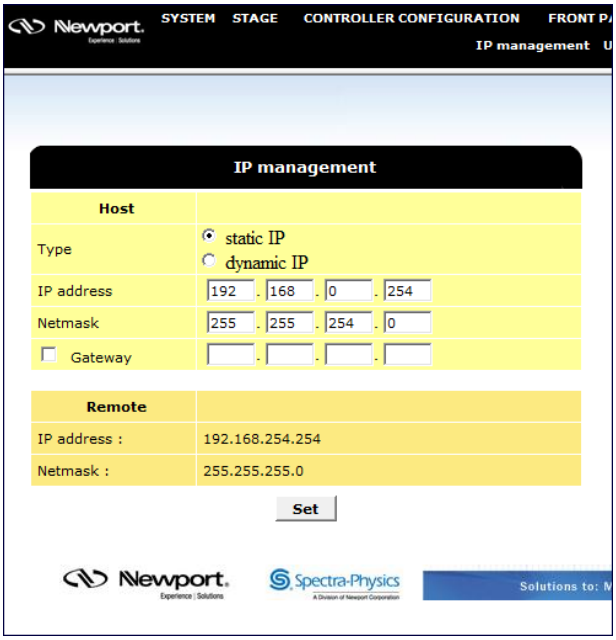
If you don't want to connect the XPS controller through a Corporate Network you may skip to section 3.7: "Connecting the Stages".

NOTE

If you want to change the IP address of the XPS controller, follow the explanation in the next section. It is necessary to keep using the gray cross-over Ethernet cable to connect the XPS controller directly to the PC.

3.5.4 Connecting the XPS to a Corporate Network using Static IP Configuration

Once you are logged in using the previously described steps for direct connection, you can change the IP configuration of the controller in order to connect the XPS over a Network. Select "CONTROLLER CONFIGURATION" of the web-site and select the sub-menu "IP Management".



The static IP address, the subnet mask and the Gateway IP address must be provided by your Network Administrator to avoid network conflicts. Once you have these addresses, you can input them in the IP configuration window as shown above. The above shown addresses are only examples.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

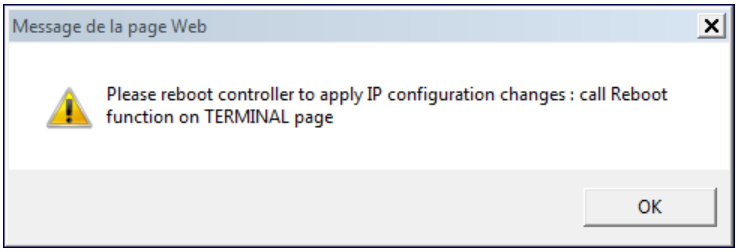
NOTE

To avoid conflict with the REMOTE Ethernet plug, the IP address must be different from 192.168.254…

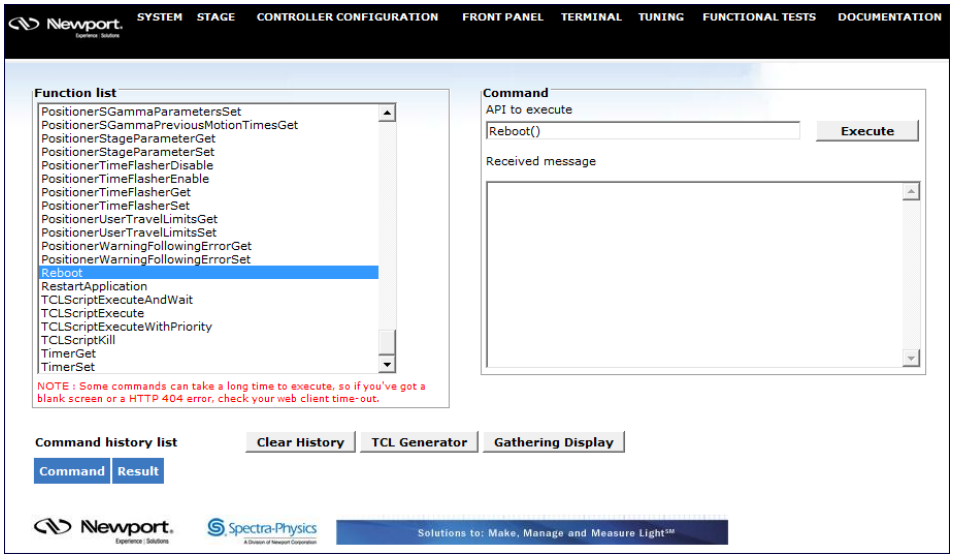
NOTE

For the majority of Networks, the setting above for the Subnet Mask will work. However, for larger networks (200 computers or more), the Subnet Mask address must be verified with the IT department. In most cases and for larger networks, the Subnet Mask is set to 255.255.0.0.

Once the appropriate addresses for the Static IP configuration are set, click on SET and the following screen appears:



Go to the TERMINAL window, and double click on Reboot function, then press the Execute button:



Wait for the end of the boot sequence. There is an initial beep a few seconds after power on and a second beep when the controller has finished booting. The time between the first beep and the second beep is approx. 12-18 seconds.

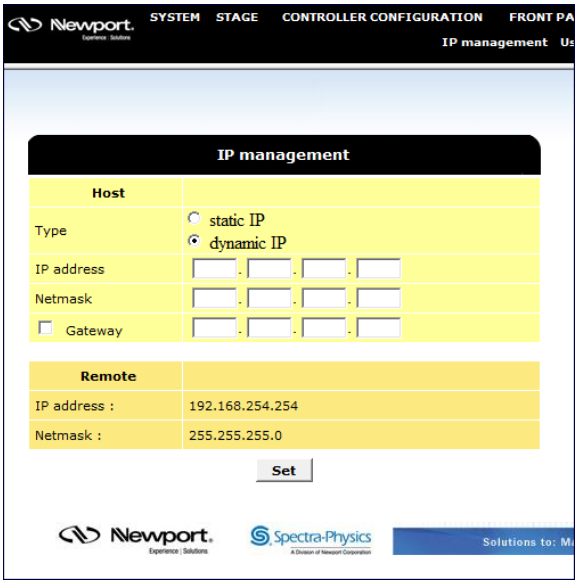
Connect the CAT-5 network cable (black) to the HOST connector of the XPS controller and to your network.

After restarting the controller and restoring your PC's Ethernet card default configuration, open the Internet browser and connect using your given Static IP address.

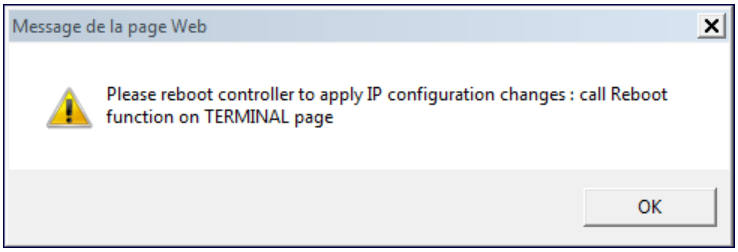
If you don't want to connect directly to the Corporate Network using the Dynamic IP Configuration, skip to section 3.7: "Connecting the Stages".

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

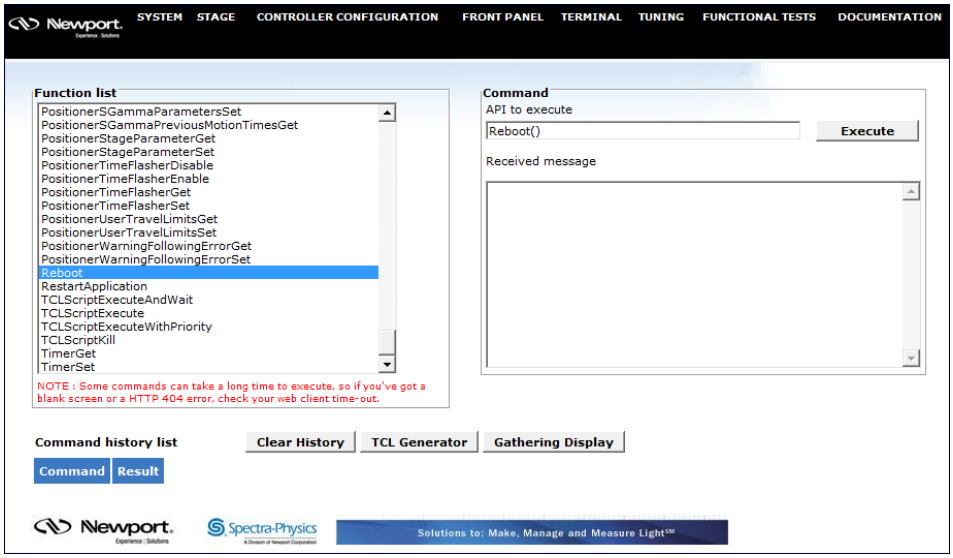
3.5.5 Connecting the XPS to a Corporate Network using Dynamic IP Configuration



Click the SET button and the following screen appears:



Go to the TERMINAL window, double click on the Reboot function, then press the Execute button:



Make sure that the standard CAT-5 network cable (black) is connected to the HOST connector of the XPS controller and to your network.

After restarting the controller, open the internet browser and connect using your controller name. Skip to section 3.7: "Connecting the Stages".

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE

Do not use Dynamic IP configuration if your DHCP server uses Windows NT 4.0 server.

3.5.6 Recovering a lost IP configuration

If you want to recover a lost IP configuration, you need to connect the PC directly to the REMOTE connector at the back of the XPS with the gray cross-over cable.

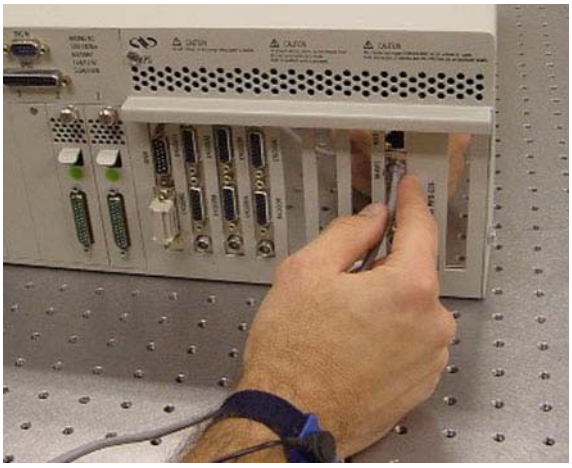
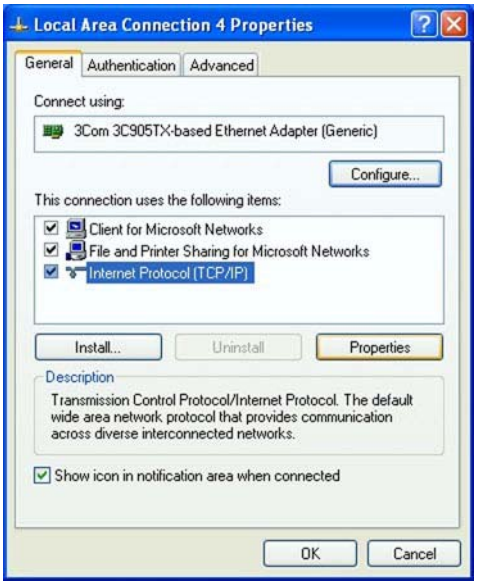


Figure 15: Direct connection to the XPS using a cross-over cable and the REMOTE connector.

First, the IP address on the PC's Ethernet card must be set to match the fixed IP address of the XPS in the REMOTE plug (192.168.254.254). Following is the procedure to set the Ethernet card address.

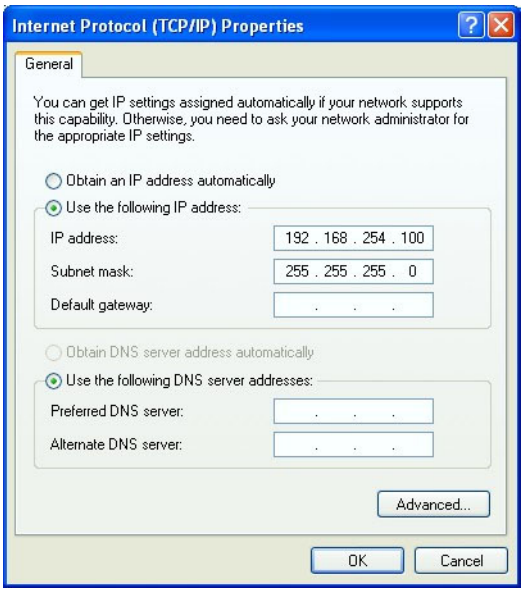
This procedure is for the Windows XP operating system (almost similar process to Windows 7):

1. Start Button > Control Panel > Network Connections (Network and Sharing).
2. Right Click on Local Area Connection Icon and select Properties.



3. Highlight Internet Protocol (TCP/IP, TCP/IP4) and click on Properties.
4. Type the following IP address and Subnet Mask as shown in the next figure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



5. Click "OK".

NOTE

The last number of the IP address must be set to any number between 2 to 253: 100 in this example.

NOTE

When configuring the controller to be on the network, the settings for the PC's Ethernet card must be set back to default which is "Obtain an IP address automatically".

Once the Ethernet card address is set, you are ready to connect to the XPS controller. Following is the procedure for connecting to the controller:

6. Open Internet Browser and connect to **http://192.168.254.254**

Login:

Name: **Administrator**

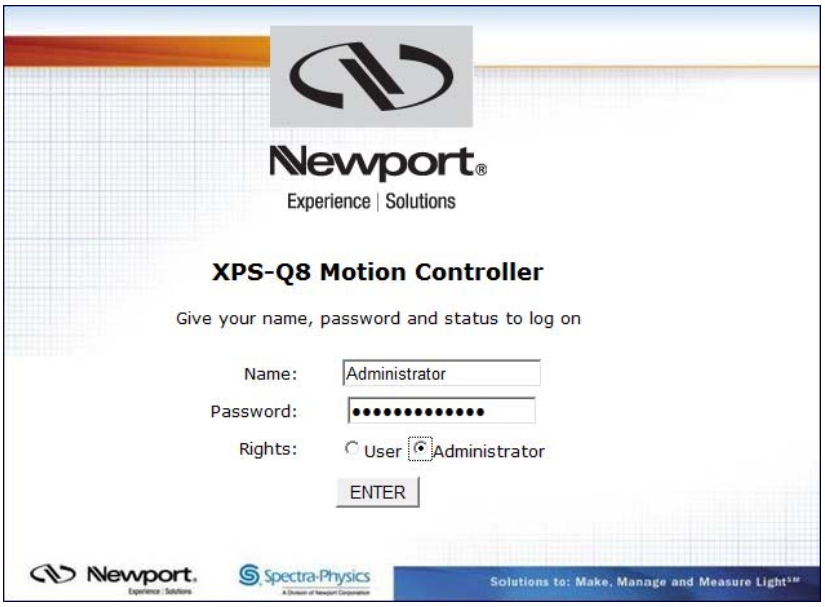
Password: **Administrator** (Please see the picture below).

Rights: **Administrator**

NOTE

Please note that the login text is case sensitive.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



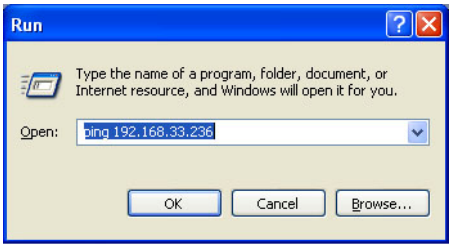
Once you are logged in, you can change the IP configuration by following the steps described in section 3.5.4 or 3.5.5 depending on your configuration.

NOTE

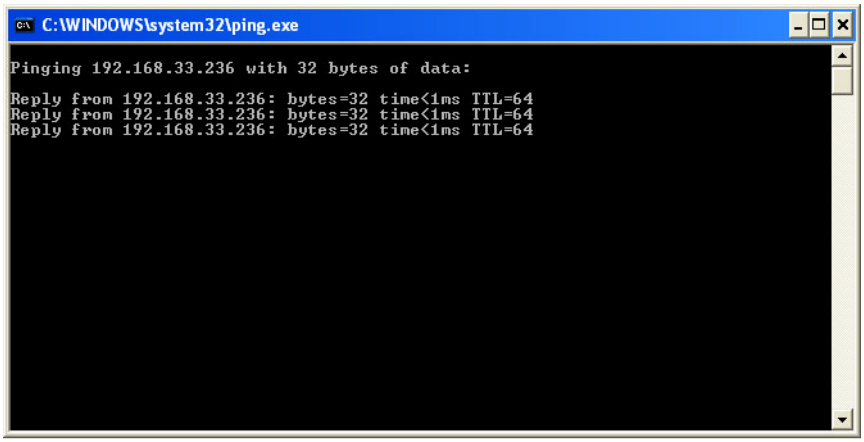
If you want to reset the IP address to the default factory setting, follow the section 3.5.4 to set the IP address back to 192.168.0.254.

3.6 Testing your XPS-PC Connection and Communication

To check if the XPS communicates with to the host computer, send a ping message from the computer to the XPS. This is done through the Windows menu: Start->Run->, then type: ping + IP address of the XPS. See the example below for the IP address 192.168.33.236:



If the XPS is connected and communicates properly, it replies in the terminal window that appears after clicking on the OK button:



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If the XPS controller is not communicating, the window displays that the time delay of the request is exceeded. Ensure that the correct cable and IP addresses are set properly.

3.7 Connecting the Stages



CAUTION

Never connect/disconnect stages while the XPS controller is powered on.



CAUTION

Mount the stage(s) on a flat, stable surface before connecting to the XPS controller.

With the power off, carefully connect the supplied cables to the stage and to the appropriate axis connector at the rear of the controller. Secure both connections with the locking thumbscrews.

When using stages with an analog encoder interface, a separate encoder cable must be connected to the corresponding axis connector of the control board labeled "Encoder 1" to "Encoder 8".

Please note that the XPS controller will not detect cross-connection errors between the motor of one stage and the encoder of another stage. Make sure that motor, encoder and other cables are plugged to the appropriate axis driver card and encoder connectors.

CAUTION



It is strongly recommended that the user read section 3.4: "System Setup" before attempting to turn the controller on. Serious damage could occur if the system is not properly configured.

All Newport ESP-compatible stages are electrically and physically compatible with the XPS controller. ESP-compatible stages are visually identified with a blue "ESP Compatible" sticker on the stage. If an ESP-compatible motion system was purchased, all necessary hardware to connect the stage with the XPS controller is included. The stage connects to the XPS via a shielded custom cable that carries all the power and control signals (encoder, limits, and home signals). The cable is terminated with a standard 25-pin D-Sub connector.

"Dummy stages" might be used to simulate a stage. These allow users to configure and test the system's behavior without having real stages connected.

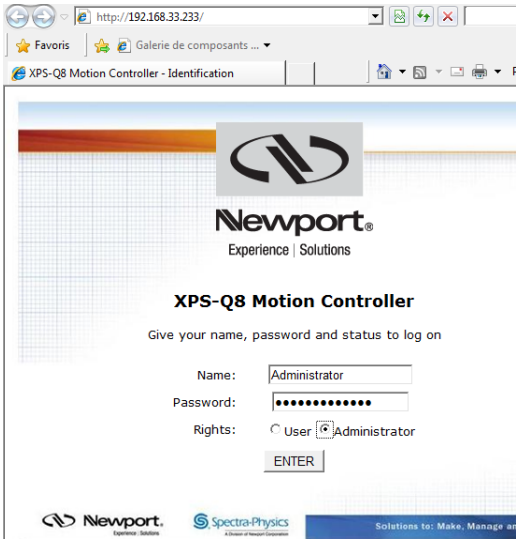
For a dummy stage, use a male 25-pin D-Sub connector with the signals for + and - travel limits connected to ground, and plug this connector to the Newport stage interface (see pinout description of the motor driver connectors in appendix F). Configure your system with a number of these dummy stages. Dummy stages can be found in the stages.ini file (see Admin/Config folder of the controller) under [DUMMY_STAGE].

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.8 Configuring the Controller

When the driver boards are installed and the IP address is configured, the controller can be configured for the stages:

- Switch off the XPS controller.
- Connect the stages or motion devices.
- Switch on the XPS controller and wait for the end of the boot sequence. There is an initial beep a few seconds after power on and a second beep when the controller has finished booting. The time between the first beep and the second beep is approx. 12-18 seconds.
- Open an internet browser and connect to http://<your fixed IP address>



Login: **Administrator**

Password: **Administrator**

Rights: **Administrator**

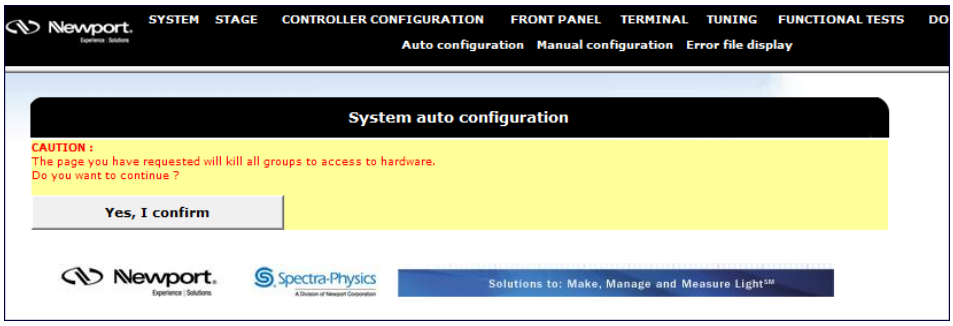
There are two possibilities to configure the controller: Auto configuration and manual configuration. Auto Configuration is the simplest method to configure the controller, but has some limitations:

- Auto configuration works only with Newport ESP compatible positioners.
- Auto configuration configures all detected positioners as single axis groups. However, single axis groups provide limited functionality (no synchronized motion, no trajectories, no XY or XYZ compensation). To take full benefit of the capabilities of the XPS controller, a manual configuration is needed.
- For non-Newport stages or very old Newport stages, manual configuration is required. See document ConfigurationWizard.pdf for details. This document is accessible from the XPS web tools under the tab DOCUMENTATION.
- Manual configuration is also required for some vacuum compatible stages (no ESP chip) and for stages with adjustable home position (-1, 0, +1), if the home position is changed from the standard position 0 to -1 or +1. The positions +1 and -1 require different settings in the stage data base, as the home switch position is not recognized by the ESP chip.

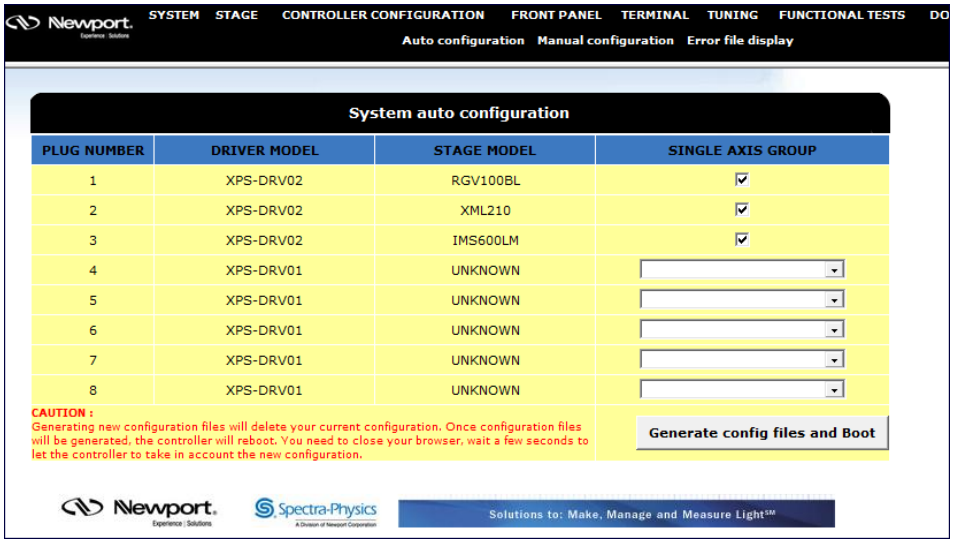
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.8.1 Auto Configuration

When logged in as Administrator, select **SYSTEM**, then **"Auto configuration"**. The following screen appears:



If you want to continue, click the "Yes, I confirm" button and the following page appears:



Check, if all connected stages are recognized by the system. If yes, click "GENERATE CONFIGURATION FILES".

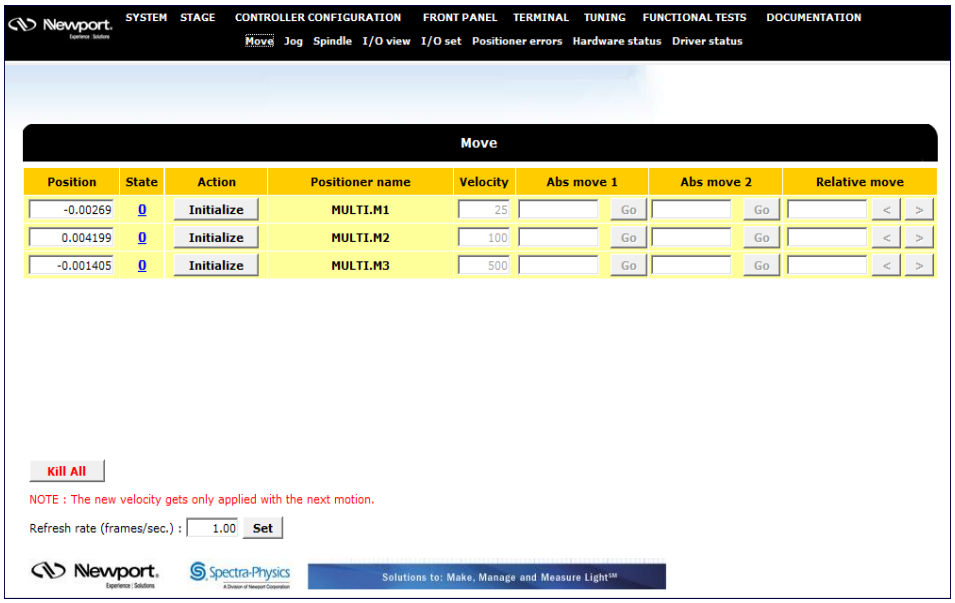
The controller reboots and the following screen appears (this may take up to 16 seconds):



Click "OK".

When the controller has finished booting (a second beep after 12-18 seconds), press "F5" to reload the page, select **FRONT PANEL**, and then select **"Move"**. The following screen appears:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Click "Initialize". The State number changes from 0 to 42 and the Action button changes from "Initialize" to "Home". Click "Home". The stage starts moving to find its reference position. When done, the state number is 11 and the action button changes to disable. Enter an allowed position value in the "Abs move 1" field and click "Go". The stage moves to this absolute position.

Your system is now ready to use. For more advanced functions, please read the rest of this manual.

NOTE

In "AUTO-CONFIGURATION" the default group is set as SingleAxis. To set the positioners to a different group type, use manual configuration.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.8.2 Manual Configuration for Newport Positioners

Manual configuration provides users access to all capabilities of the XPS controller.

For manual configuration, users first need to build the stage data base using the web tool "**Add from Database**" under the main tab **STAGE**. When adding a new stage from this web tool, the controller copies the parameters from its internal database (which contains parameters for all Newport stages) and stores these parameters in a file called stages.ini. Hence, the stages.ini file contains the parameters for only a subset of stages as defined by the user. Users can assign any name for their stages. The default name is the Newport part number, but in some cases it makes sense to use a different name. This way, for instance, it is possible to add the same set of parameters several times in the stage data base under different stage names. Later, you can modify certain parameters, like travel ranges or PID settings, to optimize the stage for different applications.

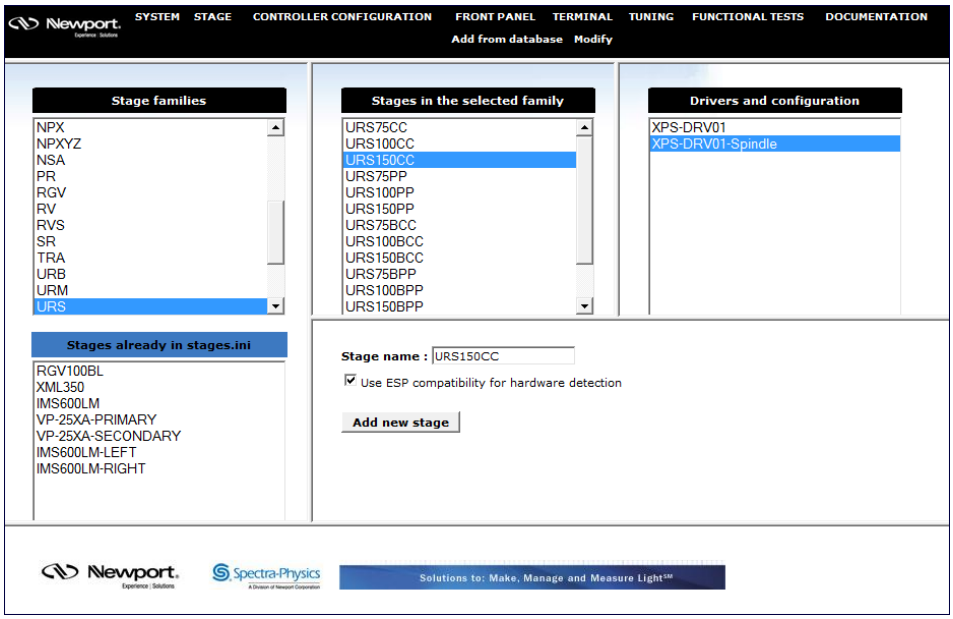
All stage parameters can be modified using the Web Tool **"Modify"** under the main tab **STAGE.** Alternatively, the stage parameters can be modified directly in the stages.ini file using a text editor. The stages.ini file is located in the Config folder of the XPS controller. This folder is accessible via ftp, see chapter 5 for details.

When all stages are added to the stages.ini file, build the system using the web tool "**Manual Configuration**" under the main tab **SYSTEM**. In this tool, the stages get assigned to positioners and the positioners get assigned to motion groups. Please refer to chapter 6.3 for details on the different motion groups and their specific features. The group name and positioner name can be any user given name. Once the system has been built, all system information is stored in a file called system.ini. Also, the system.ini file is located in the Config folder of the XPS controller.

The following describes the different steps needed to add a stage, to modify the stage parameters and to build a manual configuration. Chapter 4.0 provides further information about some of the steps described here.

Once you are logged in as Administrator, click on **STAGE** and then click on **"Add from database"**.

1. The following screen appears:



2. Double click to select a family name from the list.

3. Double click to select the part number corresponding to your hardware.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4. Select the driver (corresponding to your hardware) and configuration.

   For all continuous rotation stages, you can choose between a "regular" stage configuration and a "Spindle" configuration. A Spindle is a specific rotary device with a periodic position reset at 360° (by default), meaning 360° equals 0°. When defining the stage as Spindle in the stages.ini, you must assign this stage also to a Spindle group in the system configuration and vice versa. For details about Spindles, please refer to section 6.3.

5. Once the stage name appears, you can modify it as needed(see comments above).

6. The box "Use ESP Compatibility for Hardware detection" is checked by default. If your stage has an ESP chip inside (see the ESP-compatible sticker on the stage) this box should remain checked. Otherwise, with vacuum compatible stages or with old Newport stages, or with non-Newport stages, uncheck this box.

7. Click on "Add new stage" to add the stage.

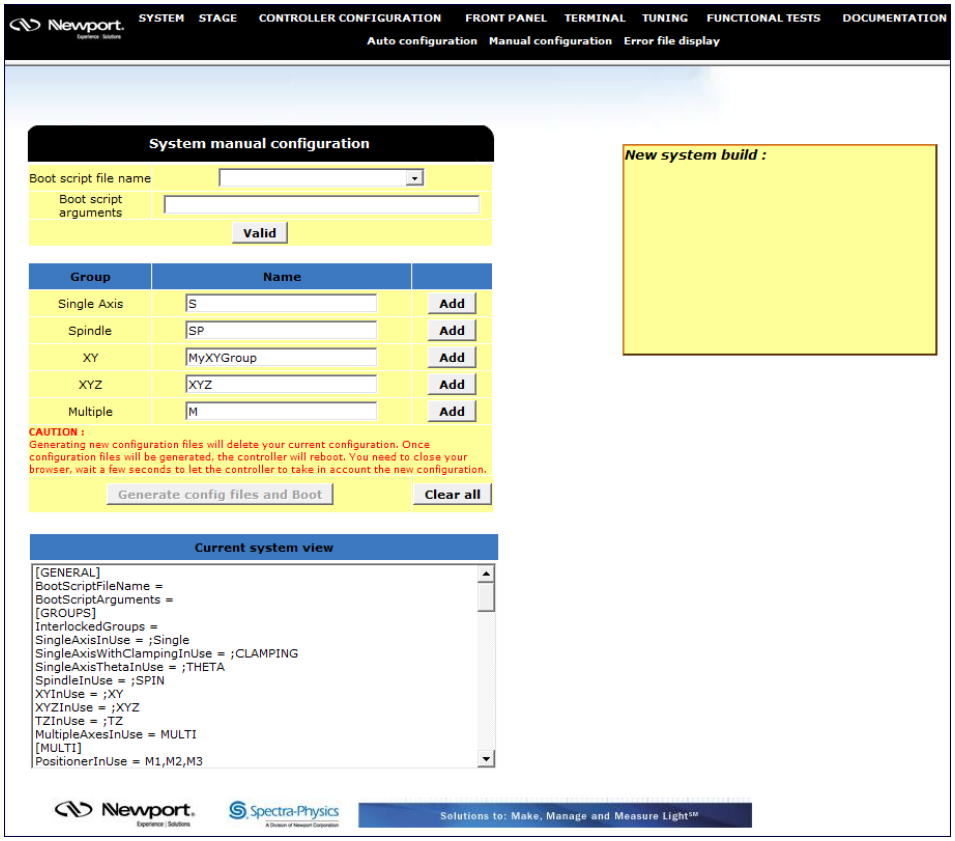
   Once all stages have been added, you can review or modify these parameters from the screen **"Modify"** under the main tab **STAGE**.

NOTE

From this screen, you have access to all stage parameters. Only experienced users should modify these parameters. For the exact meaning of the different parameters, please refer to the document ConfigurationWizard.pdf, accessible from the main tab DOCUMENTATION.

---

8. When finished with all stages, click on **"Manual Configuration"** under **SYSTEM**. The following screen appears:

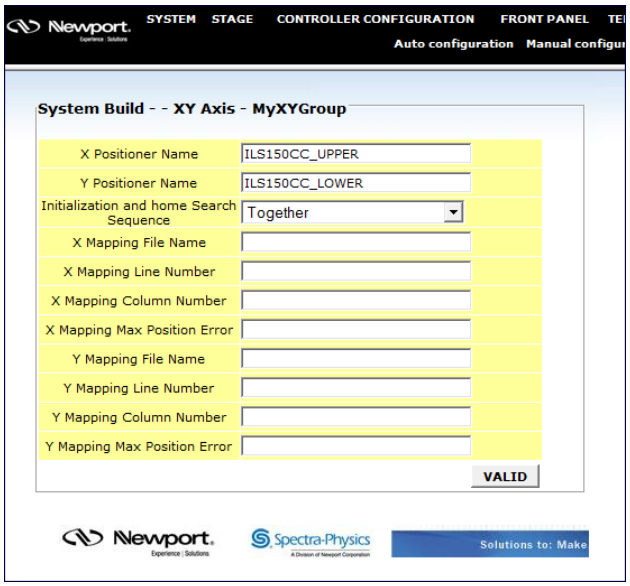


9. Enter a group name.

   For example, if you are setting up two ILS stages, you can set them up as two Single Axis groups, one XY group or one or two MulipleAxis groups. Any group name can be given. In the example the name of the XY group is MyXYGroup.

10. Click on "ADD" to get to the next screen:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

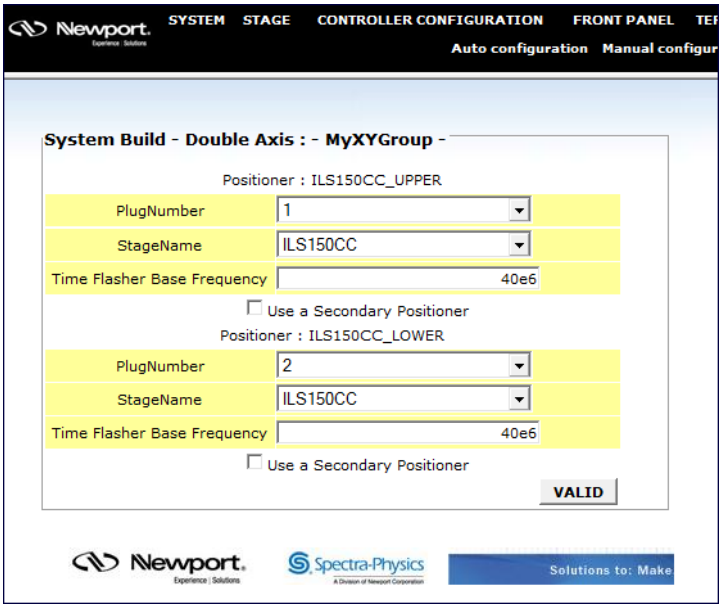


11. Enter the positioner names.

Any positioner name can be used. In this example the X positioner name is ILS150CC_UPPER. The home sequence can be either "Together" or "X then Y".

The other fields refer to the error compensation (mapping) of the XPS controller, see chapter 10.0 for details. For the first configuration, don't enter anything in these fields.

12. Click on "VALID" to get following screen:



13. Enter the appropriate PlugNumber. The plug number is the axis number where the stage is physically connected to the XPS controller. Looking at the rear of the controller, plug number 1 is the first plug on the right and the number increases to the left.

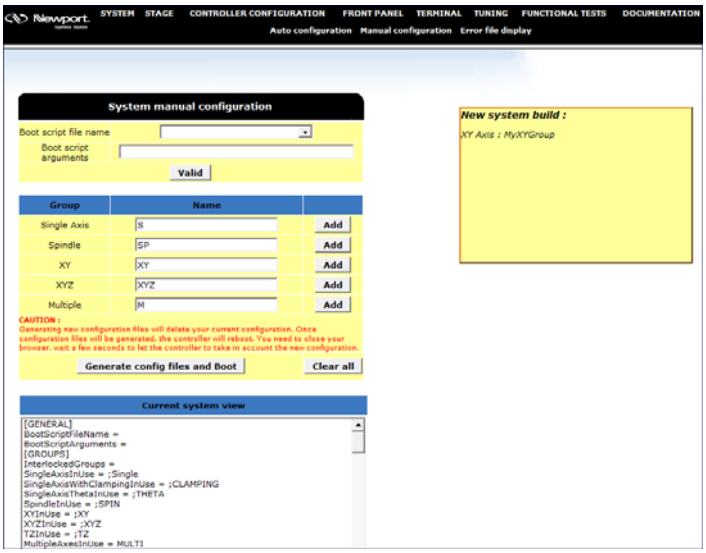
14. Select the StageName from the list of stages. These stage names refer to the stages defined with the Web Tool "Stage Management".

15. Specify the Time Flash Base Frequency value, the default is $40e^6$ (must be between 39.5 e^6 and 40.5 e^6 Hz).

16. Checking the box "Use a secondary Positioner" assigns a secondary positioner for a gantry configuration. For details about gantries, please refer to section 4.9. Don't check this box for a regular XY group or for a regular SingleAxis group.

17. Click on "VALID" to return to the initial screen.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



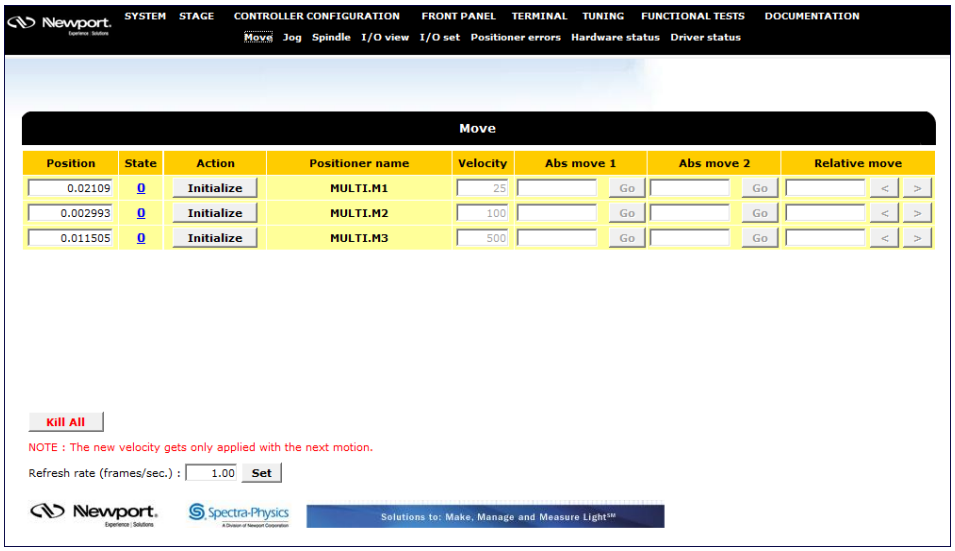
18. Continue the same way with the other motion groups.

19. When done, click on "Create new system.ini file" to complete the System configuration. The controller re-boots and the following message appears:



Click on "OK".

When the controller has finished booting (a second beep after 12-18 seconds), press "F5" to reload the page, select **FRONT PANEL**, then select **"Move"**. The following screen appears (Group names will be different according to your definition):



Click "Initialize". The State number changes from 0 to 42 and the Action button changes from "Initialize" to "Home". Click "Home". The stage starts moving to find its reference position. When done, the state number is 11 and the action button is "Disable". Enter an allowed position value in the "Abs move 1" field and click "Go". The stage moves to this absolute position.

Your system is now ready to use. For more advanced functions, please read the rest of this manual.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.8.3 Manual Configuration for non Newport stages

For configuring the XPS controller to stages or positioning devices not made by Newport, use the tool **"Add Custom Stage"** under the main tab **STAGE**. For detailed information about this tool, please refer to the document ConfigurationWizard.pdf provided under the main tab **DOCUMENTATION**.

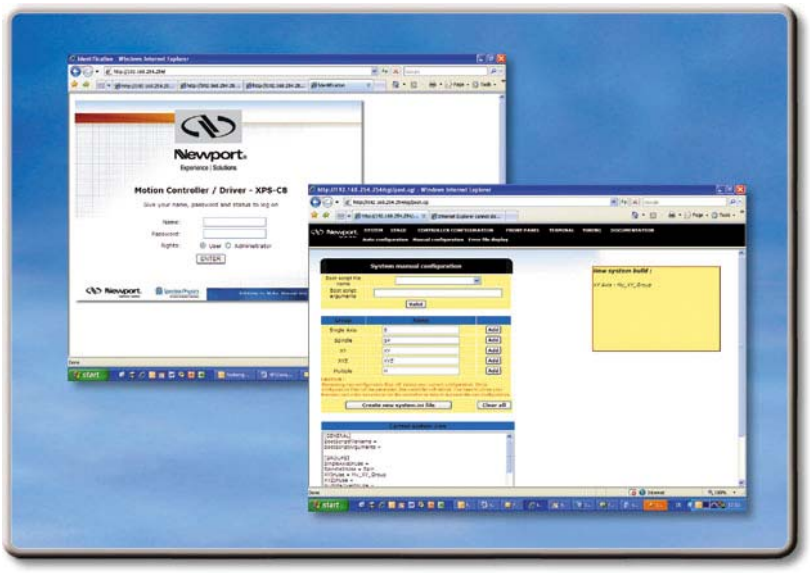
3.9 System Shut-Down

To shut down the system entirely, perform the following procedure:

Wait for the stage(s) to complete their moves and come to a stop.

Turn off the power using the power switch located above the power cord at the back of the controller.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Software Tools

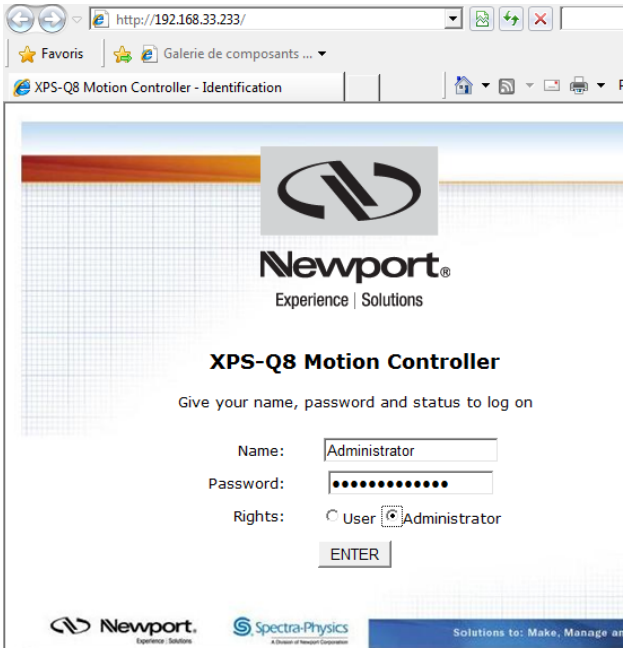
4.0 Software Tools

4.1 Software Tools Overview

The XPS software tools provide users a convenient access to the most common features and functions of the XPS controller. All software tools are implemented as a web interface. The advantage of a web interface is that it is independent from the user's operating system and doesn't require any specific software on the host PC.

There are two options to log-in to the XPS controller: as “User” or as “Administrator”. Users can log-in only with User rights. Administrators can log-in with User and with Administrator rights. When logged-in with Administrator rights, you have an extended set of tools available.

The predefined user has the log-in name **Anonymous**, Password **Anonymous**. The predefined Administrator has the log-in name **Administrator**, Password **Administrator**. Both the log-in name and the password are case sensitive.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The main tab is displayed across the top of the XPS Motion Controller/Driver main program window, and lists each primary interface option. Each interface option has its own pull-down menu that allows the user to select various options by clicking the mouse's left button.

On the following pages, a brief description of all available tools is provided.

Administrator Menus



Sub-Menu for CONTROLLER CONFIGURATION



Restricted set of User Menus



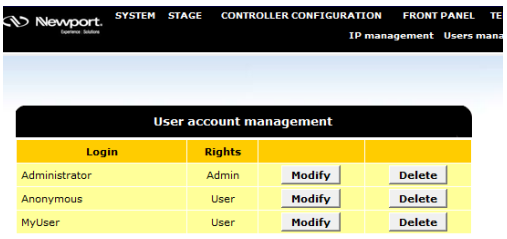
4.2 CONTROLLER CONFIGURATION – Users Management

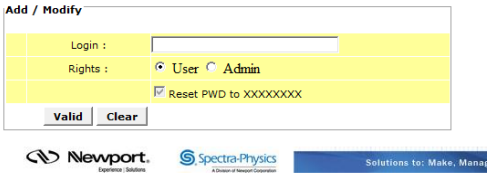
This tool allows managing User accounts. There are two types of users: Administrators and Users. Administrators have configurations rights. Users have restricted rights to use the system.

The following steps are needed to create a new user:

1. Enter a new user name in the “login” field.
2. Choose the access rights: “User” or “Admin”.
3. Check the box “Reset PWD to XXXXXXXX”:

   Your password is reset to XXXXXXXX.
4. Select the “VALID” button to add the new access account.





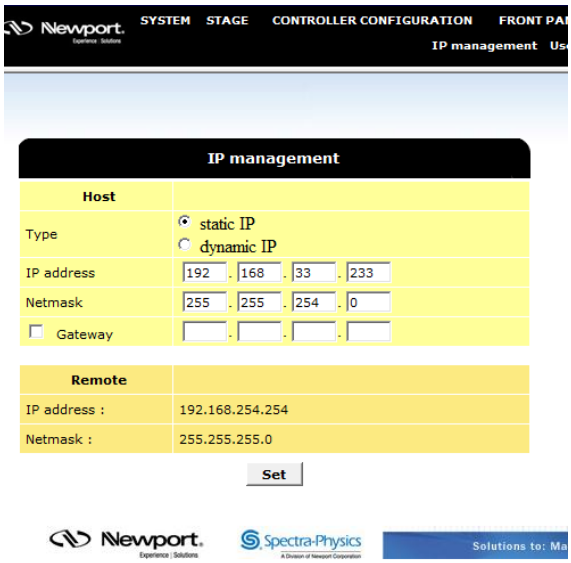
NOTE

The default password is XXXXXXXX and must be changed after the first log in.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

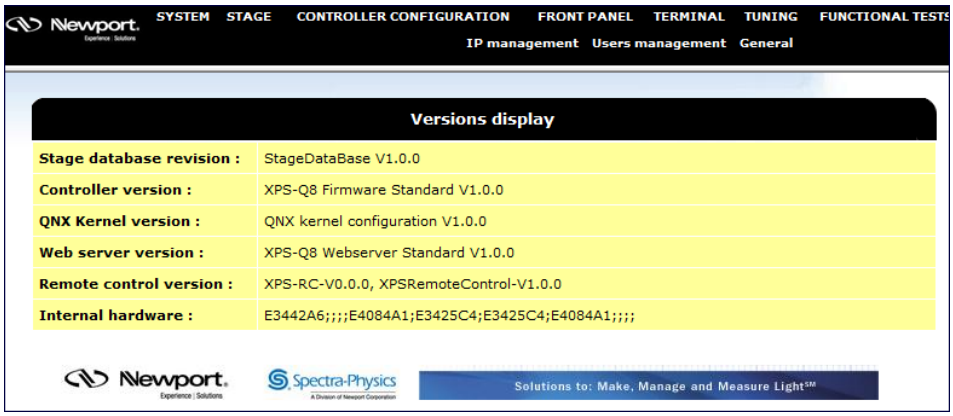
4.3 CONTROLLER CONFIGURATION – IP Management

See chapter 3.5 for details.



4.4 CONTROLLER CONFIGURATION – General

This screen provides valuable information about the firmware and the hardware of the controller. It is an important screen for troubleshooting the controller.

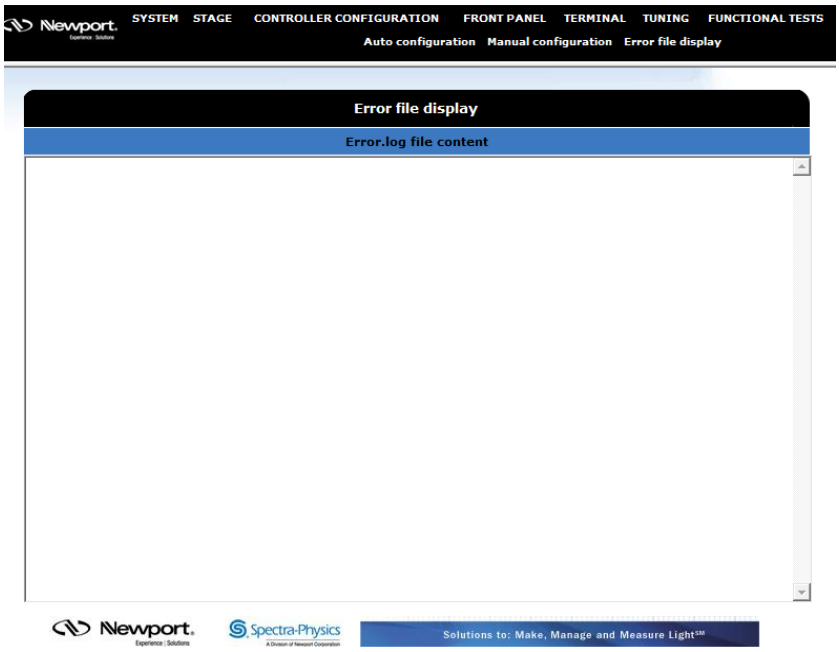


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.5 SYSTEM – Error file display

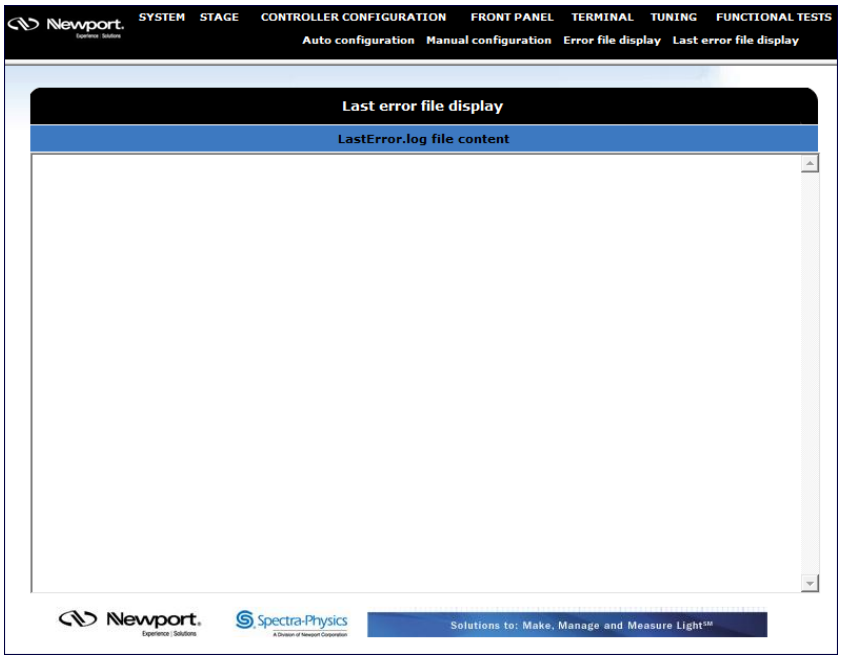
The Error File Display is another important screen for troubleshooting the XPS controller. When the XPS encounters any error during booting, for instance due to an error in the configuration files or because the configuration is not compatible with the connected hardware, there are entries in the error log file that guides you to correct the error.

When no error is detected during the system boot, this file is blank.



4.6 SYSTEM – Last error file display

The Last error file display shows errors encountered in the last XPS boot. When no error is detected during the last system boot, this file is blank.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

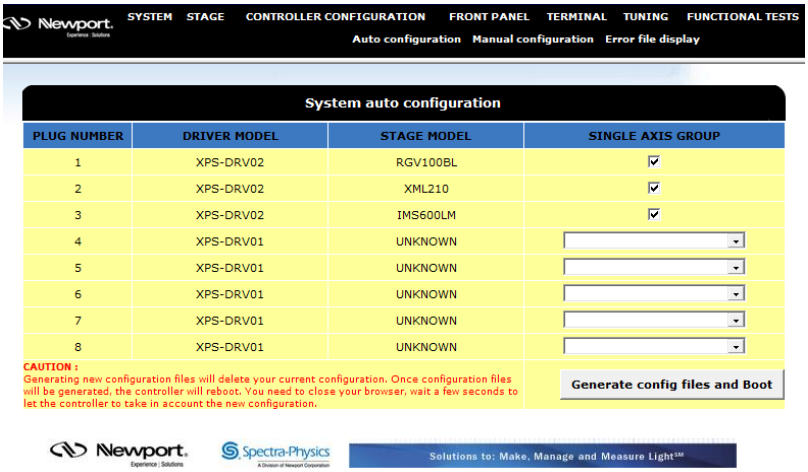
4.7 SYSTEM – Auto Configuration

With the help of this screen, a quick, basic configuration of the XPS controller can be done. Check/un-check those stage models that you want/don't want the XPS controller to configure to. When done, click "Generate Configuration Files". The XPS controller reboots. After re-booting, you are able to use the XPS controller in this basic configuration. For further information, refer to chapter 3.8.1.

NOTE

"Generate Configuration Files" deletes your current system.ini configuration file. For troubleshooting a system, make sure that you have a copy of the original system.ini file for recovery.

Under Driver Model and Stage Model, all motor drivers and Newport ESP compatible stages seen by the XPS controller are listed. This screen also provides valuable information for diagnosing or troubleshooting the system.



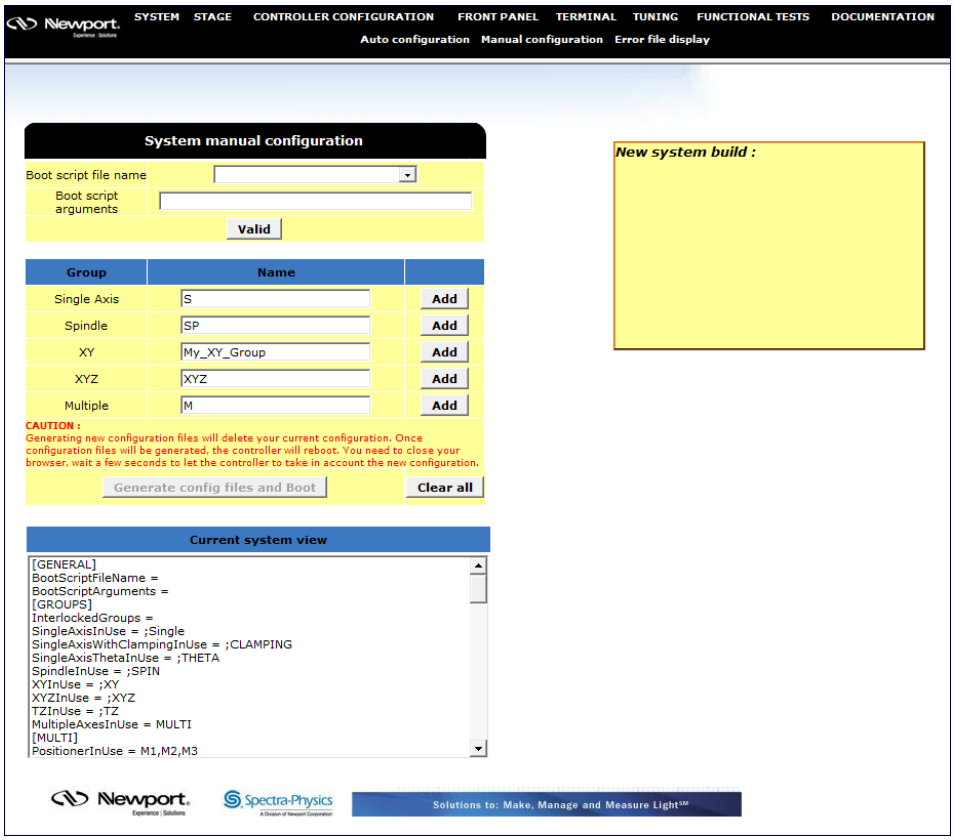
4.8 SYSTEM – Manual Configuration

Manual Configuration allows you to review the current system configuration or to define a new one. See also chapter 3.8.2 for further information.

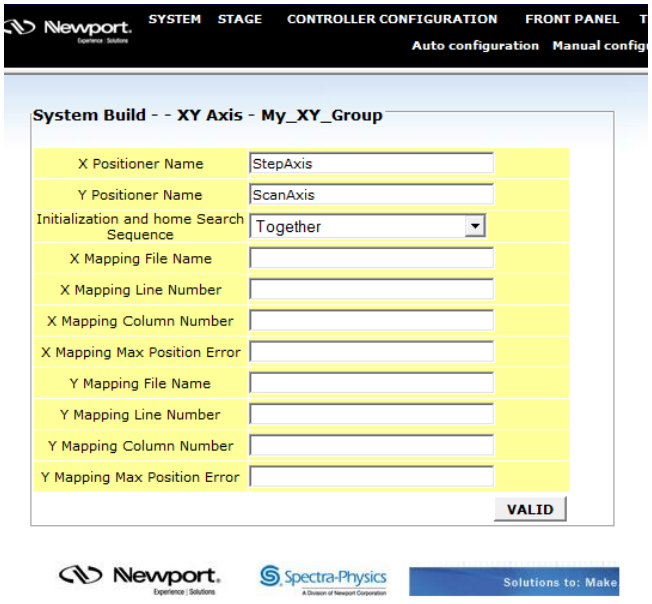
To create a new system configuration, define all motion groups that should belong to that system. It is not possible to append a motion group to an existing configuration from this tool. To define a new motion group, do the following:

1. Enter the name of the new group (My_XY_Group in this case). Click on "ADD" to confirm the new group.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



2. Enter the name for each positioner associated with the motion group (StepAxis and ScanAxis in this case). Define the home sequence (“Together” or “XThenY” or “YThenX”). For error compensation, define the name and structure of the correction data, otherwise leave these fields blank. For details about error compensation, see chapter 10.0. When done, click on “VALID” to accept the configuration.

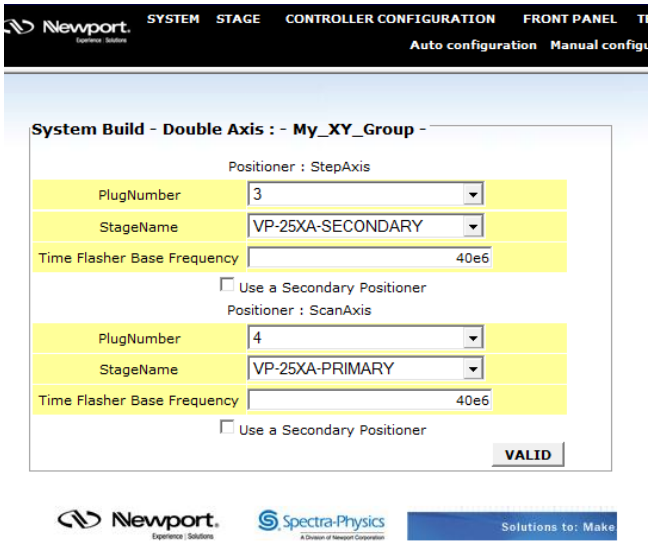


3. Specify the plug number. The plug number is the number of the drive card (1 to 8) where the stage is physically connected to the XPS controller (see back of XPS controller). Select the name of the Stage from the stage data base (scroll down menu).
   Checking the box “Use a secondary Positioner” assigns a secondary positioner for a gantry configuration. A gantry is a motion device where two positioners, each of them having a motor, an encoder, limits, etc., are used for a motion in one direction.

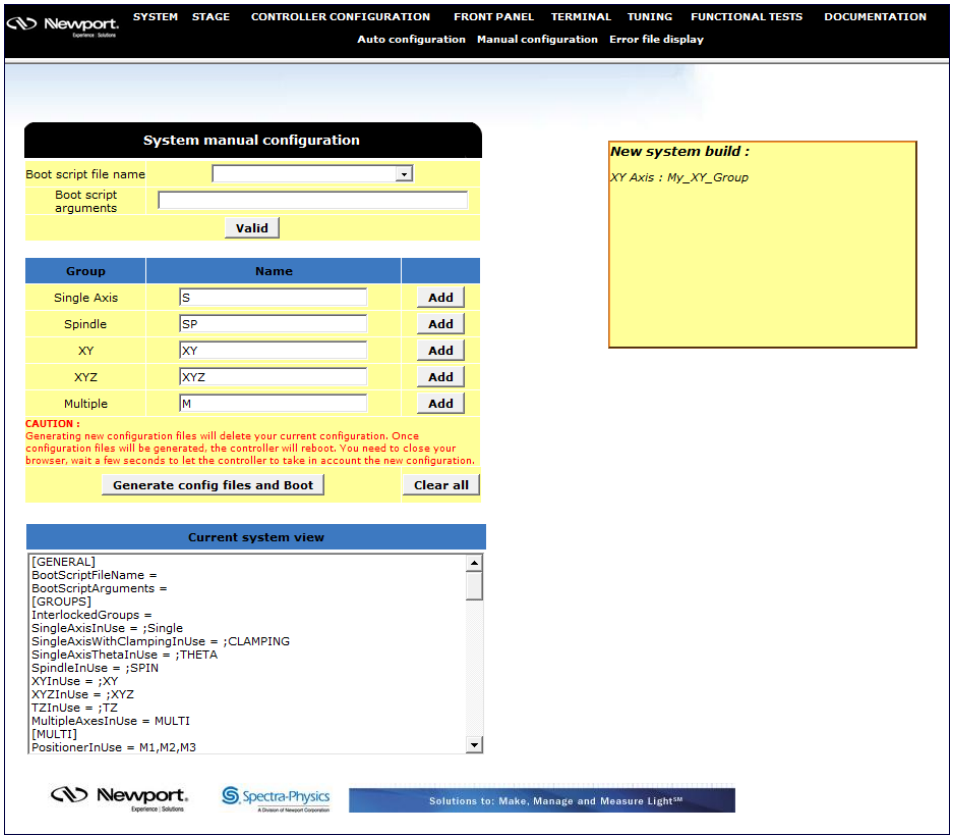
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Like most gantries, the two positioners are rigidly attached to each other. Hence, all motions, including motor initialization, homing, and emergency stops must be done in perfect synchronization. For details about Gantries and their configuration, please refer to section 4.9.

Specify the Time Flash Base Frequency value, default is 40E6 (must be between 39.5E6 and 40.5E6 Hz).
When all positioners are configured, click on “VALID” to confirm the group configuration.



4. When the configuration of each positioner is validated, the new group is listed in the “New system build” window.



5. Do the same for all other motion groups. When done, click on “Generate config files and Boot” to apply the new configuration.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE

"Generate config files and Boot" deletes the current system.ini file. To create a copy of the current system.ini file, retrieve this from the "..admin\config" folder of the XPS controller.

The following screen appears:



Click on "OK".

6. When the controller has finished booting (second beep after 12-18 seconds), select the **SYSTEM** tab, then **"Error File Display"**. When there is no entry in the error file, your system is configured correctly and ready to use. If not, this file provides some valuable information for troubleshooting; see also chapter 4.5.

This is an example of a system.ini file with one XY group and one Spindle group:

```
[GENERAL]
BootScriptFileName =
BootScriptArguments =

[GROUPS]
SingleAxisInUse =
SpindleInUse =  Spin
XYInUse =  My_XY_Group
XYZInUse =
MultipleAxesInUse =

 [My_XY_Group]
PositionerInUse = StepAxis,ScanAxis
InitializationAndHomeSearchSequence = Together
;--- Mapping XY
XMappingFileName =
YMappingFileName =

 [My_XY_Group.StepAxis]
PlugNumber = 3
StageName = VP-25XA-SECONDARY
[My_XY_Group.ScanAxis]
PlugNumber = 4
StageName = VP-25XA-PRIMARY

 [Spin]
PositionerInUse = Rot

 [Spin.Rot]
PlugNumber = 2
StageName = URS100CC_Spindle
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.9 SYSTEM – Manual Configuration – Gantries (Secondary Positioners)

This section is for experienced users of the XPS controller and addresses the configuration of a gantry via a secondary positioner.

A gantry is a motion device where two positioners, each of them having a motor, an encoder, limits, etc., are used for a motion in one direction. Like most gantries, the two positioners are rigidly attached to each other, see example below. Hence, all motions, including motor initialization, homing, and emergency stops must be done in perfect synchronization.

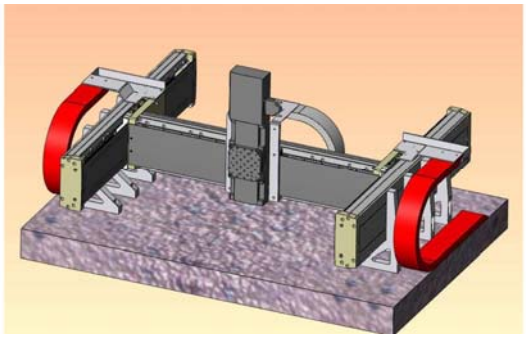
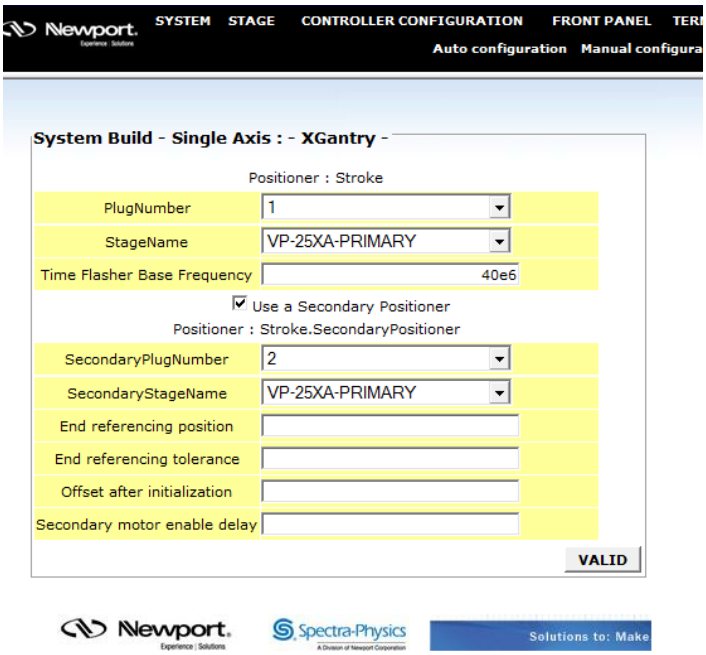


Figure 16: Example of a gantry.

The XPS controller allows configuring single axis gantries (Xx configuration) and XY gantries. For XY gantries, it is possible to define XxY, XYy and XxYy configurations. Here, X and Y refer to the primary positioner and x and y to an assigned secondary positioner.

To define a gantry, check the box “Use a secondary positioner” during the definition of a Single Axis group or XY group. See chapter 4.8 for further instructions on how to define a new motion group. When done, the following screen appears (example Single Axis group):



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Define the plug number for the secondary positioner and the name from the stage data base. The secondary positioner must have common values with the primary positioner for the following parameters:

- MaximumVelocity
- MaximumAcceleration
- HomeSearchMaximumVelocity
- HomeSearchMaximumAcceleration
- MinimumTargetPosition
- MaximumTargetPositioner

The parameters “End referencing position” and “End referencing tolerance” refer to the homing process of the gantry, see chapter 4.9.1 for details.

The parameter “Offset after initialization” is relevant only for gantries with linear motors. See chapter 4.9.2 for details. For all other gantries, enter 0 for this parameter.

Furthermore, for certain XY gantries, it is also possible to apply a variable force ratio for the two X positioners. This variable force ratio accounts for the different forces required by the primary and the secondary X-axes positioners depending on the position of the Y axis to ensure a torque-free motion. For details, see chapter 4.9.3.

NOTE

When using the gantry configuration, the secondary positioner is almost invisible in the application. All functions are sent directly to the motion group or to the (primary) positioner of that group. However, it is possible to get information about the secondary positioner by data gathering and using “SecondaryPositioner” as the positioner name. Example:

MySingleGantry.S1.SecondaryPositioner.FollowingError

For further details about data gathering, see chapter 12.0.

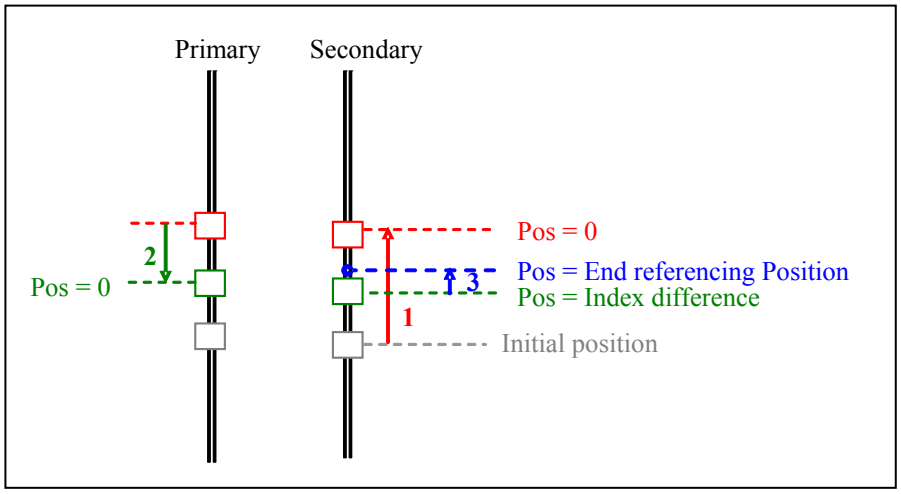
4.9.1 Home search of gantries

During the home search of a gantry, first, the secondary positioner is homed and the primary positioner follows the motion. Then, the primary positioner is homed and the secondary positioner follows the motion. At the end, the primary positioner is at its home position, but the secondary positioner will be off its home position due to the tolerances in the assembly of the gantry. This "ideal" position can be defined to be the position of best orthogonality between the X and Y axis of the gantry. The parameter “End referencing position” defines the “ideal” position of the secondary positioner when the primary positioner is at its home position. The parameter “End referencing tolerance” defines the maximum allowed distance from the secondary positioner’s ideal position, when the primary positioner is at its home position.

When the actual distance is greater than the value of the “End referencing tolerance”, homing is aborted. When the actual distance is less than the value for the “End referencing tolerance”, then the secondary positioner moves to the “End referencing position” while the primary positioner stays at its home position. Hence, this parameter corrects the angle between the gantry's X and Y axes.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The sketch below illustrates this process:



1) Search home of the Secondary positioner. The primary positioner follows
2) Search home of the Primary positioner. The secondary positioner follows.
3) If the distance of the secondary positioner's position to the "End referencing position" is greater than the value for the "End referencing tolerance", homing is aborted. If not, the Secondary positioner moves to the "End referencing position" while the primary positioner stays at its home position.

The index difference refers to the difference of the secondary positioner's position when the primary positioner is at its home position to the home position of the secondary positioner. The value for the Index difference can be queried by the function PositionersEncoderIndexDifferenceGet().

When no other metrology tools are available, the following method can be used to determine a value for the "End referencing position" of an assembled gantry:

Set the value for "End referencing position" and "End referencing tolerance" to 0. Complete the configuration of your system. After reboot, initialize and home the gantry group. With high probability, the homing will fail with error -85 due to the zero value for the "End referencing tolerance". Query the index difference with the function PositionersEncoderIndexDifferenceGet(). Repeat the initialization, homing and querying of the index difference several times and build the average and the standard deviation from all values. Now, configure a new system with the same gantry. For "End referencing position", apply the average value of the index difference. For "End referencing tolerance", apply a value that is approximately equal to 6 times of the standard deviation of the index difference. Complete your configuration and reboot your system. Initialize and home the gantry group several times to confirm the gantry is working properly.

4.9.2 Gantries with linear motors

The parameter "Offset after initialization" defines the offset of magnetic tracks of the linear motors between the primary and the secondary positioners. This parameter is important to optimize the performance of a gantry with linear motors. It ensures the correct sinusoidal commutation of the two motor signals. An accurate measurement of the offset can be done only with dedicated metrology tools.

For gantries NOT driven in acceleration mode, e.g. gantries with NO linear motors, this value is set to 0.

Also, for stages driven in acceleration mode and are configured for gantries, it is recommended to "force the initialization position" using the LMI mode (Large Move Initialization). To do so, append **LMI** to the line MotorDriverInterface=AnalogSinXAcceleration**LMI** (X = 60, 90 or 120) and add a line InitializationCycleDuration=5 at the end of the section with driver command interface parameters in the stages.ini. Example:

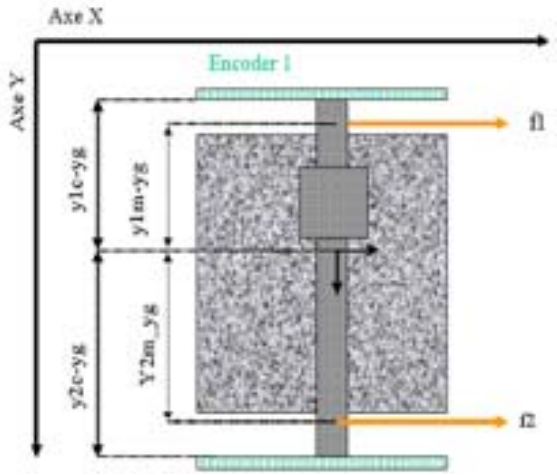
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

;--- Driver command interface parameters
*MotorDriverInterface=AnalogSin120Acceleration**LMI***
ScalingAcceleration=30641;--- units / s²
AccelerationLimit=27856;--- units / s²
MagneticTrackPeriod=24;--- units
InitializationAccelerationLevel=20;--- percent
InitializationCycleDuration=5;--- seconds

With the LMI setting, during initialization, the motor is energized and the stage moves to the closest stable magnetic position. The result is a quick motion of the stage at most by half of the length of the magnetic track. This behavior might be undesired, but provides a more failure proof method for initialization than the default initialization process, which applies only very small oscillations to the stage during initialization.

4.9.3 Gantries with linear motors and variable force ratio

For XY gantries, where the two X-axes are driven by linear motors (meaning driven in acceleration mode), it is also possible to apply a variable force ratio for the two X axes positioners. This variable force ratio accounts for the different forces required by the primary and the secondary X-axes positioner depending on the position of the Y axis. When correctly set, it ensures a torque-free acceleration and deceleration of the X-axis, see picture below for illustration.

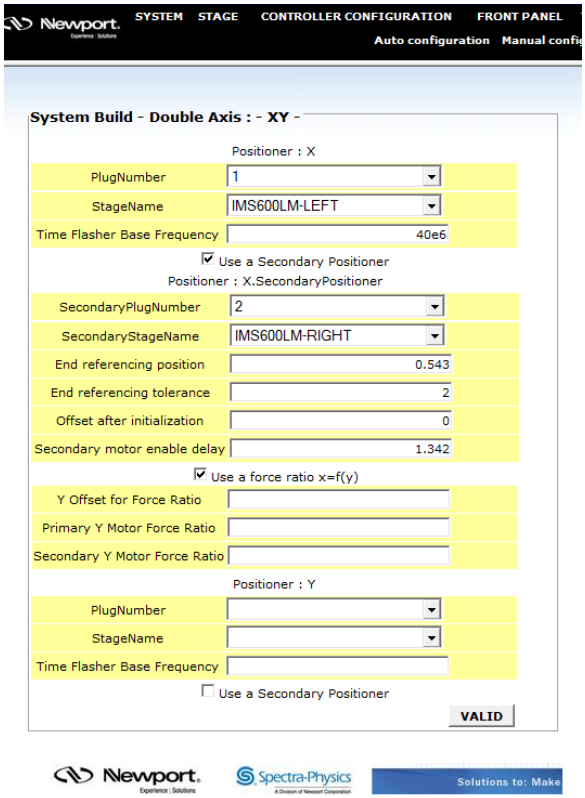


To apply a variable load ratio to an XY gantry, check the box “Use a force ratio” during the group definition. See example below. There are three parameters to input:

- Y Offset for force ratio
- Primary Y Motor Force Ratio
- Secondary Y Motor Force Ratio

A correct definition of these three parameters is not simple. For additional information about this function, please call Newport.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



This is an example of a system.ini file with one XY gantry:

```
[GROUPS]
SingleAxisInUse =
SpindleInUse =
XYInUse = MyXYGantry
XYZInUse =
MultipleAxesInUse =

[MyXYGantry]
PositionerInUse = X, Y
InitializationAndHomeSearchSequence = YThenX
XMappingFileName =
YMappingFileName =

;--- Gantry Force Ratio parameters
YOffsetForForceRatio = 0
PrimaryYForceRatio = 0
SecondaryYForceRatio = 0

[MyXYGantry.X]
PlugNumber = 1
StageName = IMS600LM

;---- Secondary positioner (X2)
SecondaryPlugNumber = 4
SecondaryStageName = IMS600LM
SecondaryPositionerGantryEndReferencingPosition = 10.2243
SecondaryPositionerGantryEndReferencingTolerance = 0.1
SecondaryPositionerGantryOffsetAfterInitialization = 7.47

[MyXYGantry.Y]
PlugNumber = 3
StageName = IMS400LM
```

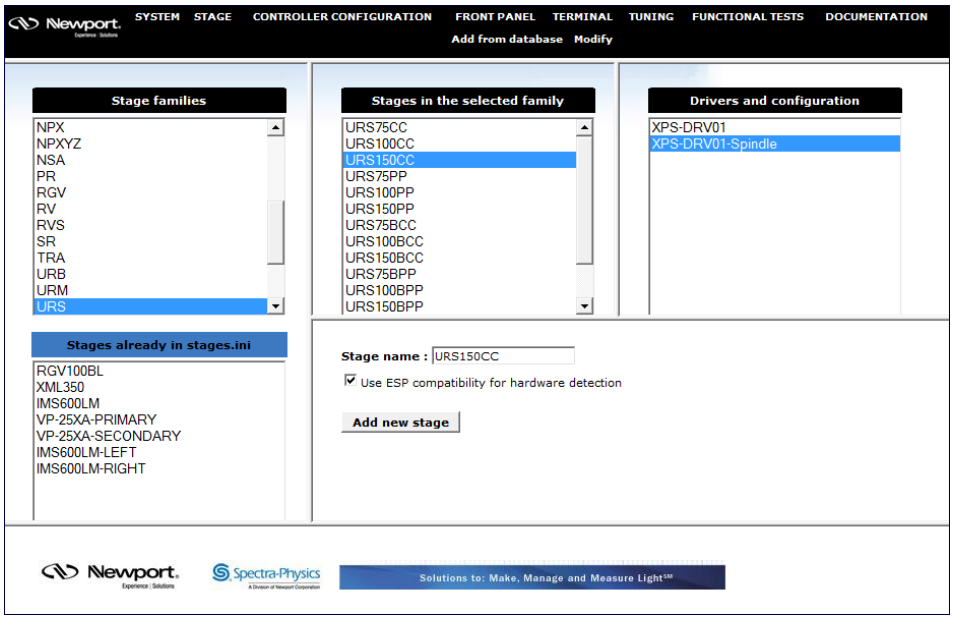

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.10 STAGE – Add from Data Base

With the help of this screen, a stage from the Newport stage data base can be added to the personal stage data base, called stages.ini. In the lower left corner, you can review the name of the stages that are already in this stage data base. To add a new stage, do the following:

1. Double click to select a family name from the list.
2. Double click to select the part number corresponding to your hardware.
3. Select the driver (corresponding to your hardware) and group configuration.

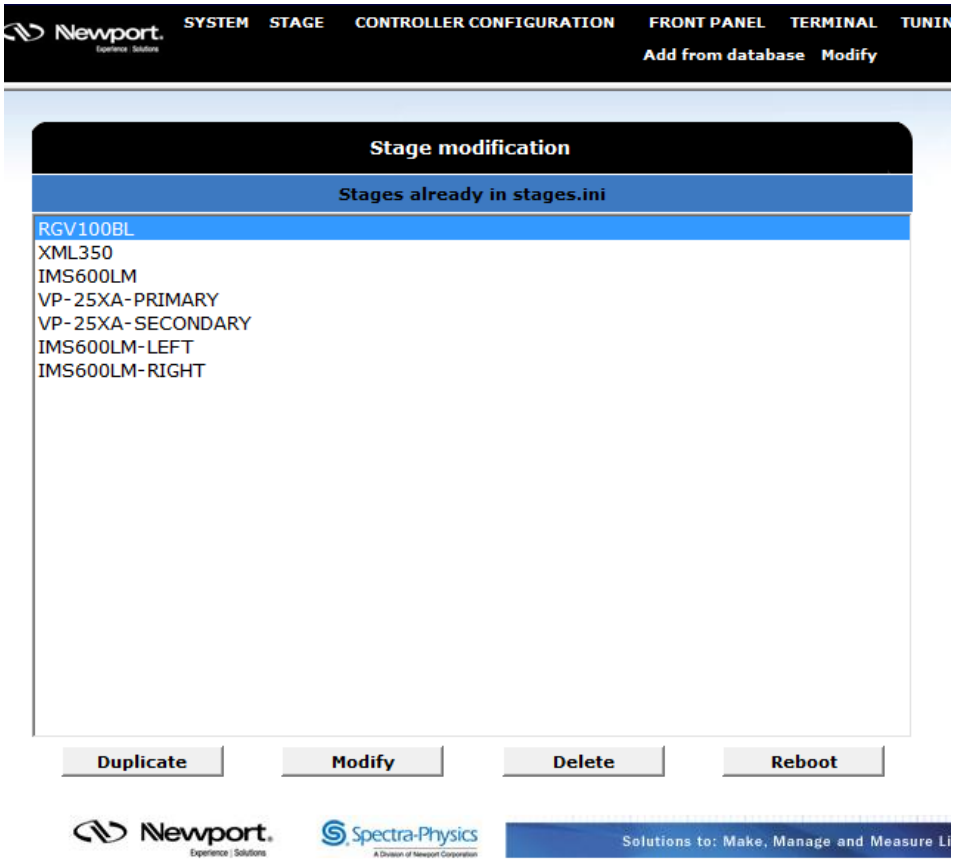
   For all continuous rotation stages, you can choose between a "regular" stage configuration and a "Spindle" configuration. A Spindle is a specific rotary device with a periodic position reset at 360°, meaning 360° equals 0°. When defining the stage as Spindle in the stages.ini, you must assign this stage also to a Spindle group in the system configuration and vice versa. For details about Spindles, please refer also to section 6.3.
4. Once the stage name appears, it can be modified. The default name is the Newport part number, but in some cases it makes sense to use a different name. This way, for instance, it is possible to add the same set of parameters several times in the stage data base under different stage names. Later, modifying certain parameters, like travel ranges or PID settings, to optimize the stage for different applications becomes straightforward.
5. The box "Use ESP Compatibility for Hardware detection" is checked by default. If the stage has an ESP chip (a blue ESP-compatible sticker is on the stage) this box shall remain checked. Otherwise, with vacuum compatible stages or with old Newport stages, uncheck this box.
6. Click on "Add new stage" to add a stage.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.11 STAGE – Modify

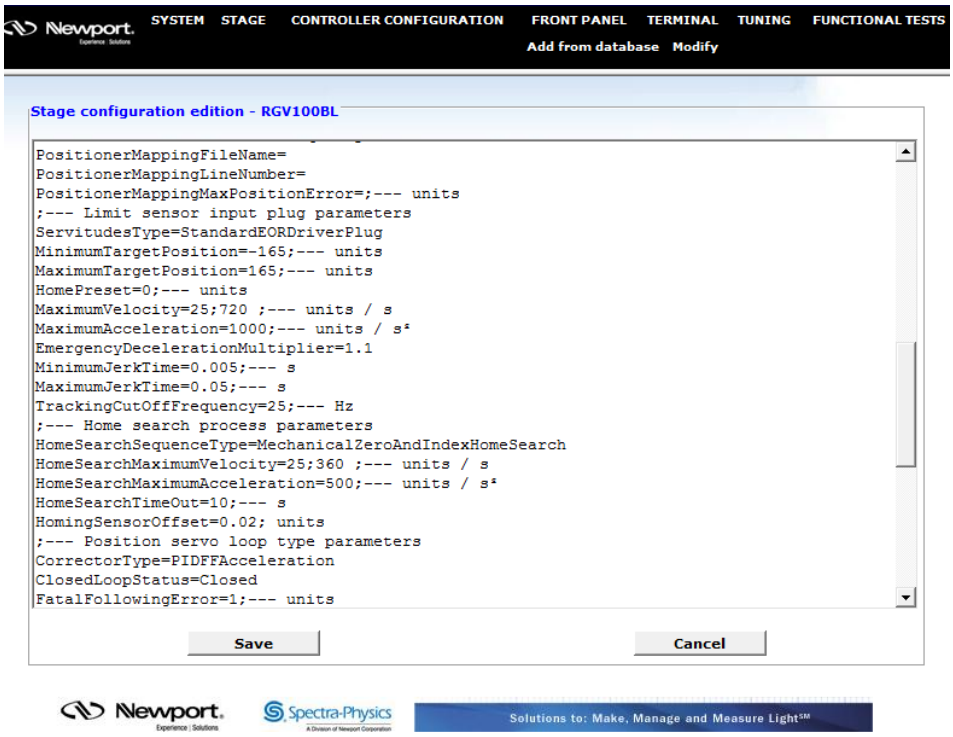
This screen allows you to review and modify all parameters of stages included in the stages.ini. Only experienced users should modify these parameters. For the exact meaning of the different parameters, please refer to the document ConfigurationWizard.pdf, accessible from the main tab **DOCUMENTATION.**



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To modify the parameters of a stage, do the following:

1. Select a stage from the list. Click on “Modify”.
2. Scroll down to the section that contains the parameters that will be modified. Parameters that require quite common changes, are the minimum and the maximum target positions of a rotation stage. For example, to enable larger rotations of a rotation stage that is not configured as a Spindle, set the maximum target position to a very high value and the minimum target position to a very low value. In this case it is also required to disable the limit switches of the rotation stage, see stage manual for details.



3. When done, click "Save" to apply the new values, or click “Cancel” if a mistake was made..
4. To take the new values into account, reboot the controller.

The same screen allows duplicating stages in the stages.ini (in most case some parameters are modified as a second step) or to delete stages from the stages.ini.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

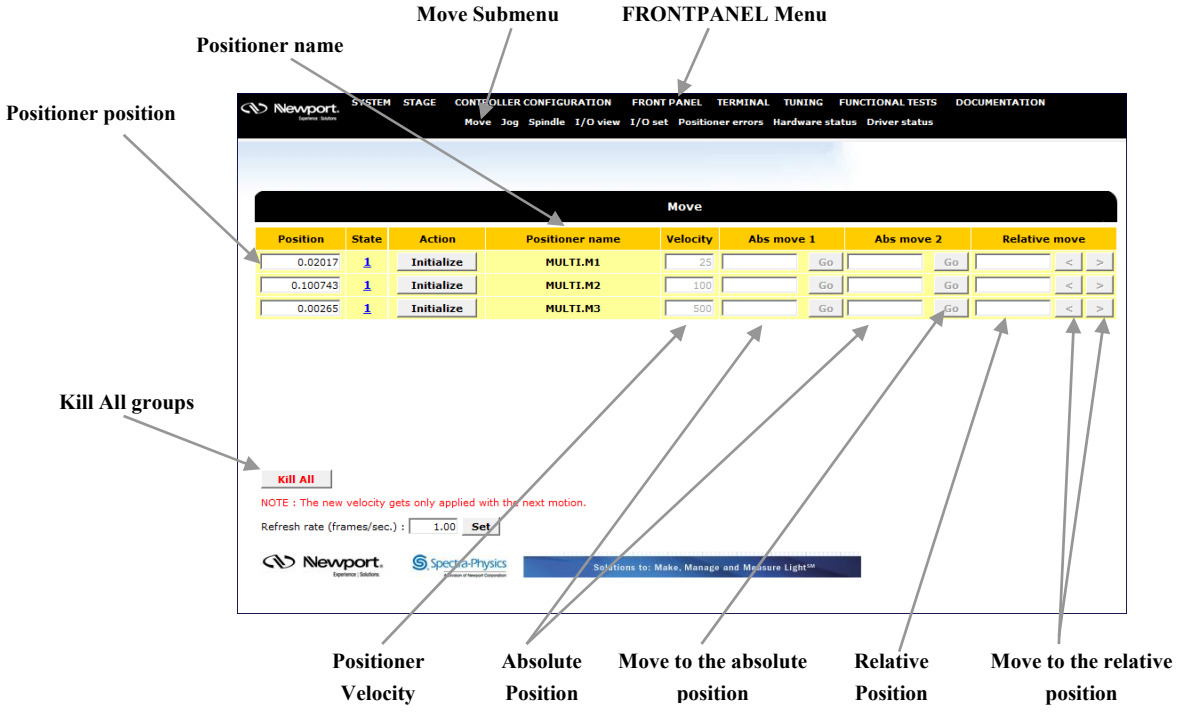
4.12 FRONT PANEL – Move

The Move page provides access to basic group functions like initialize, home, or motor disable, and executes relative and absolute moves.

The Move page also provides a convenient review of all important group information like group names, group states and positions. All motion groups are listed in the Move page.

NOTE

A spindle group can do relative moves and absolute moves. So it can be used in the Move page. See section 4.14 for more information about Spindle moves.

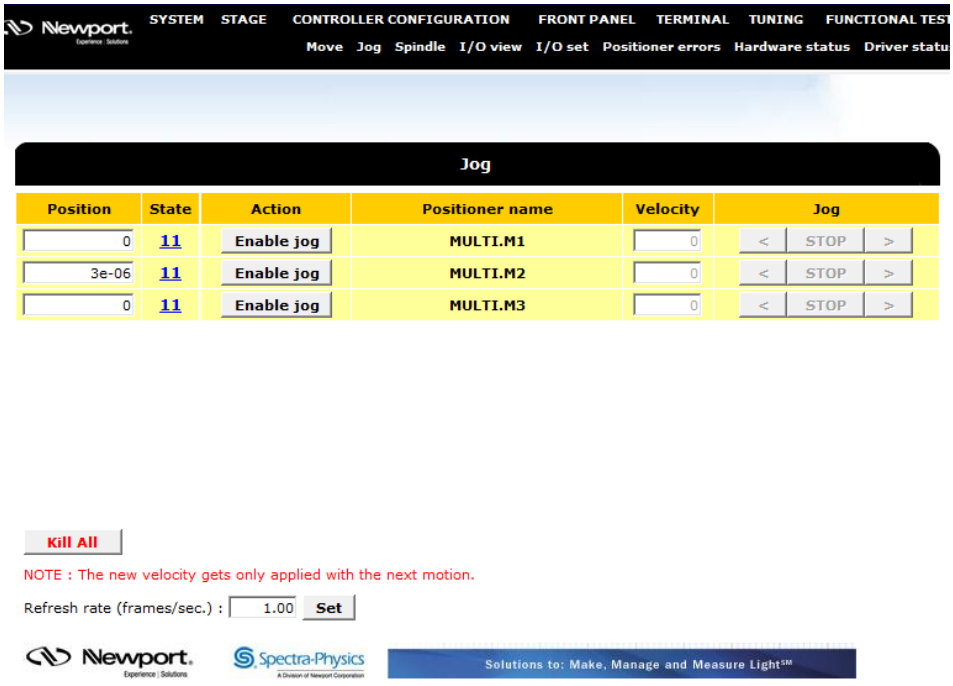


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.13 FRONT PANEL – Jog

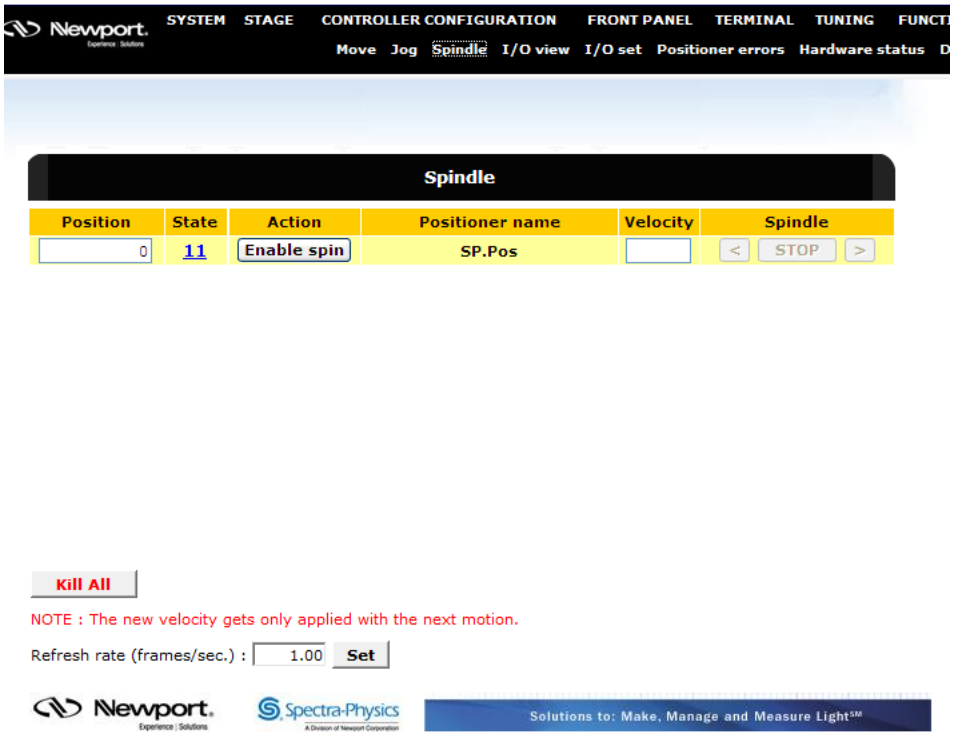
The Jog page allows executing a jog motion. A jog motion is a continuous motion, where only the speed and acceleration are defined, but no target position. Speed and acceleration can be changed during the motion (but not during the acceleration period).

For a Jog motion, the jog mode must be enabled, see “Action” button.



4.14 FRONT PANEL – Spindle

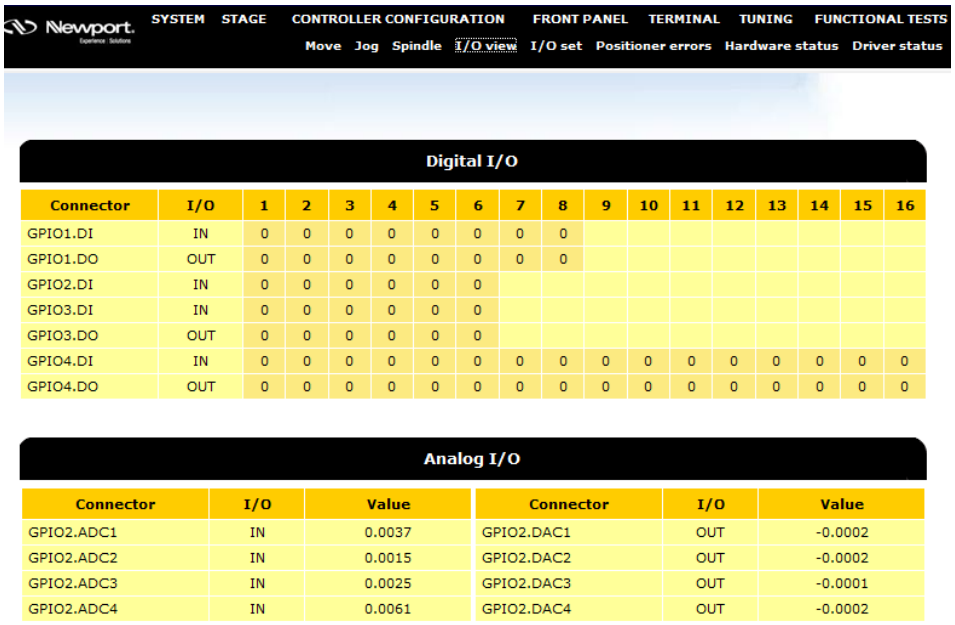
The Spindle page provides similar functions to the Jog page. However, specific jog actions are replaced by spindle actions that only work for Spindle groups.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

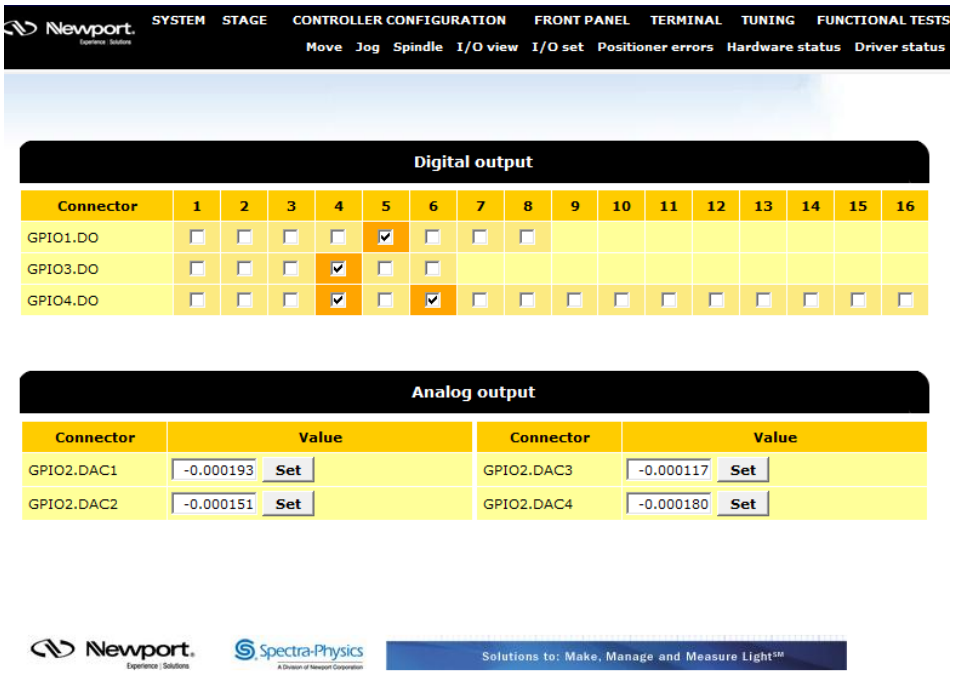
4.15 FRONT PANEL – I/O View

The I/O View page shows the current states or values of all analog and all digital I/O's of the controller. To set the outputs, use the page I/O Set.



4.16 FRONT PANEL – I/O Set

The I/O Set page allows setting the analog and digital outputs of the controller.



Newport Spectra-Physics Solutions to: Make, Manage and Measure Light℠

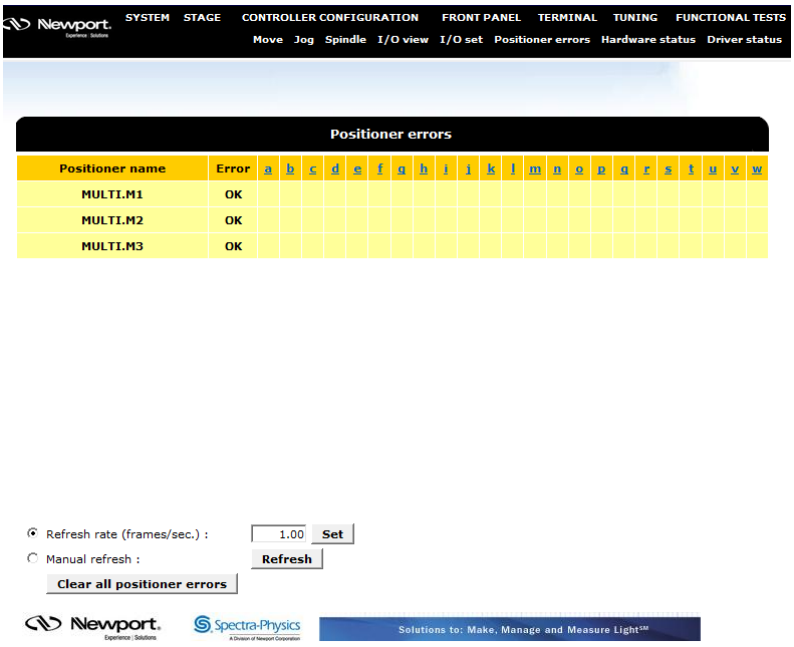
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.17 FRONT PANEL – Positioner Errors

The Positioner Errors page is an important page for trouble-shooting. When encountering any problems during the use of the system, information about the errors related to the positioners are found in this page. Hovering the cursor over the letters brings up the type of error.

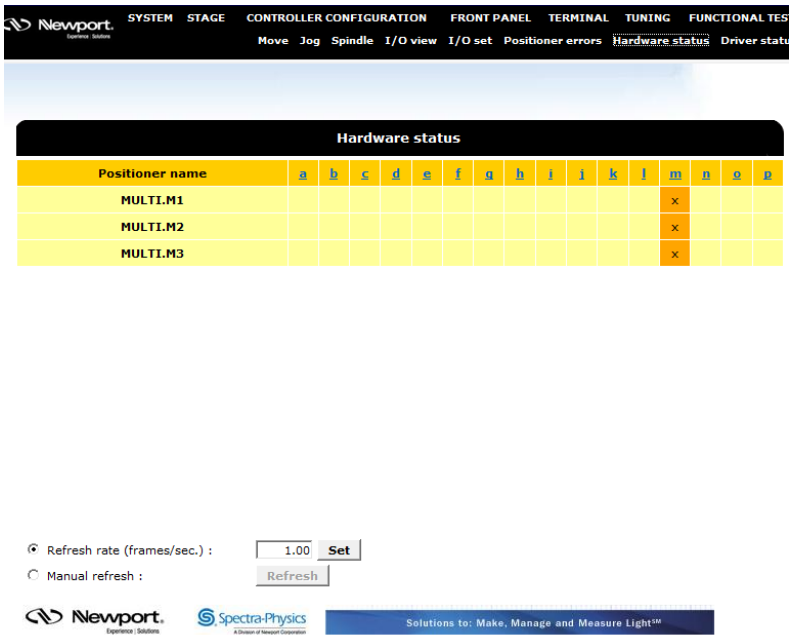
Note that all positioner errors encountered since the last “Clear all positioner errors” are displayed, even if some of the errors may no longer be present. The “Refresh” button refreshes the error page. This means that new errors will be displayed while retaining a history of previous errors..

To clear the errors, use the button “Clear all positioner errors”.



4.18 FRONT PANEL – Hardware Status

The Hardware Status page is another important page for trouble-shooting, but not all information is related to an error.

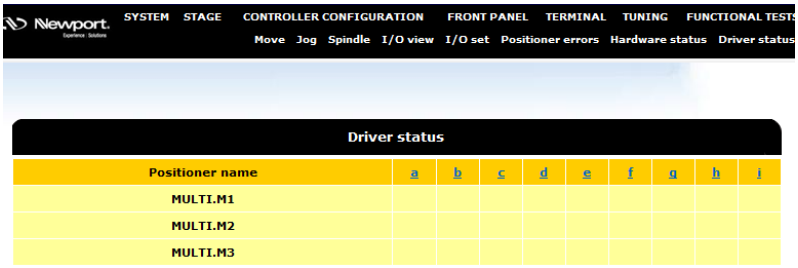


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.19 FRONT PANEL – Driver Status

The Driver Status page is another important page for trouble-shooting, but not all information is related to an error.

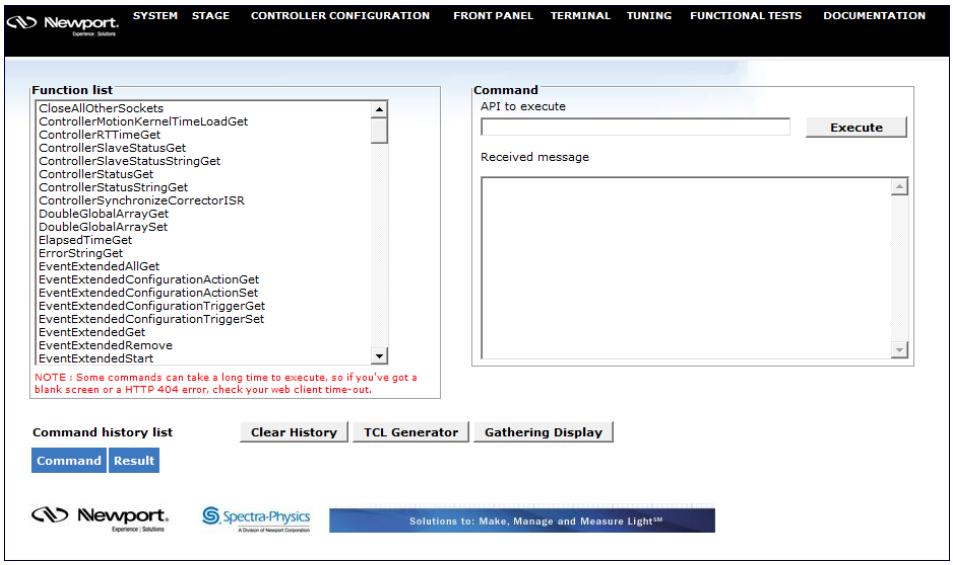
The type of status information that you can get depends on the drivers used.





4.20 TERMINAL

The Terminal screen allows the execution of all XPS controller functions. It also provides a convenient method for generating executable TCL scripts. For more details about TCL scripts, see chapter 18.1.

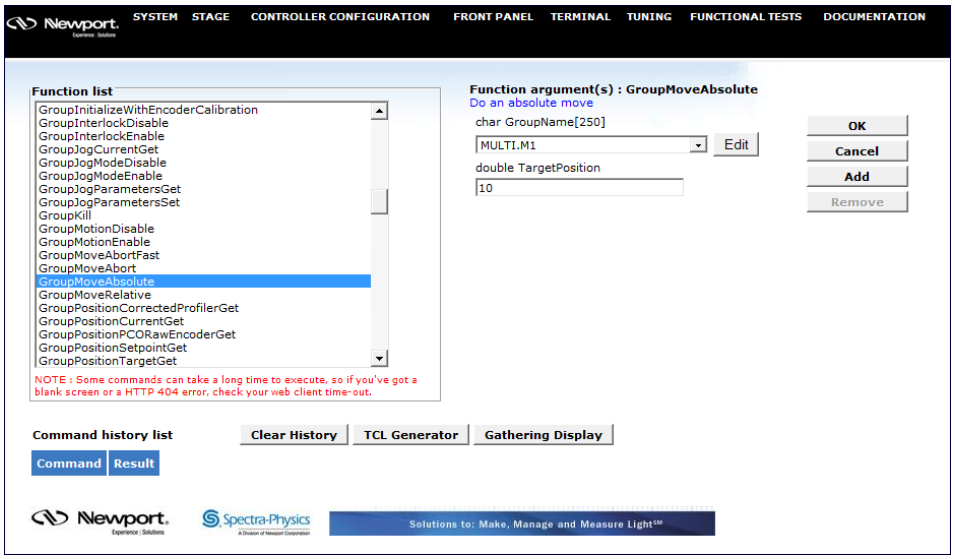


To execute a function from the Terminal, do the following:

1. Double click to select a function, which then appears in the "API to execute" window.
2. Define the arguments for the function.

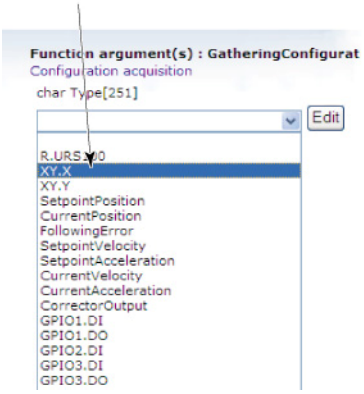
   For functions with dynamic arguments “ADD” and “REMOVE” buttons are available. Alternatively, you can use a “,” as a separator between different arguments.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

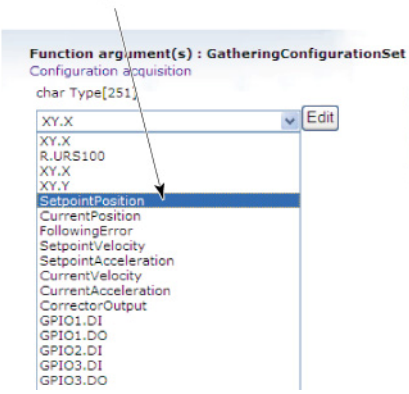


For some arguments like ExtendedEventName, ExtendedActionName or GatheringType, the argument name is not directly accessible. In these cases, define the first part of the argument name, then click in the field again and define the second part of the argument name. See the example below for defining the GatheringType with the function GatheringConfigurationSet():

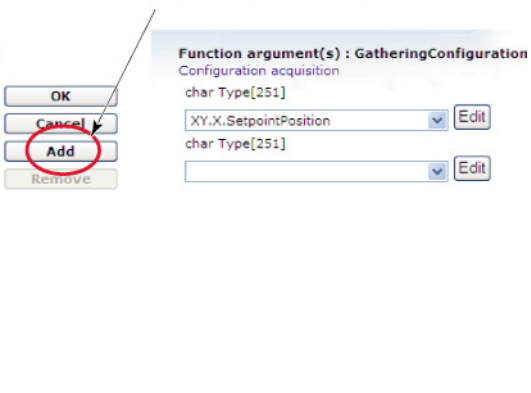
Step 1:
Select the positioner name and click.



Step 2:
Click in the choice field again.
Select parameter name and click.

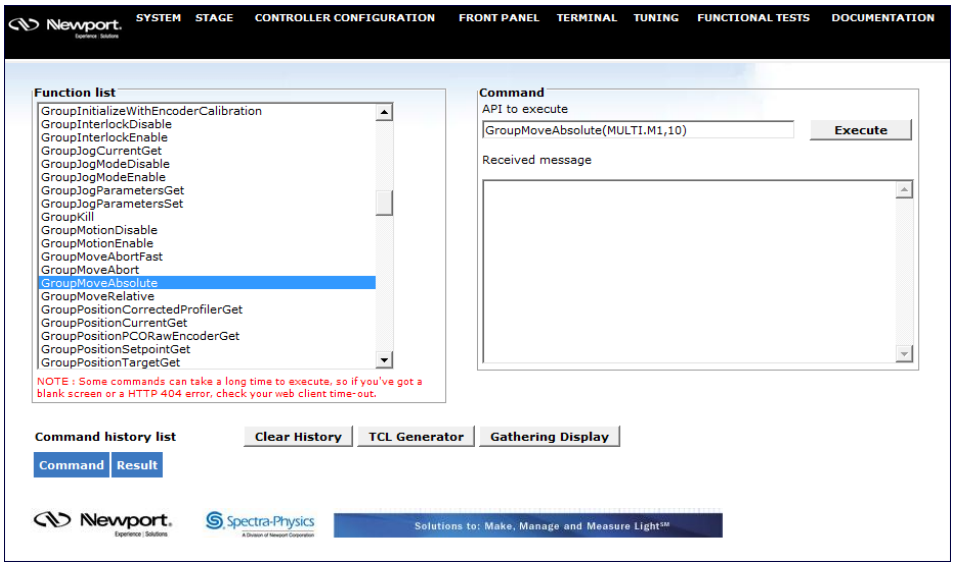


Step 3:
To add another parameter, press ADD.
Repeat step 1 and step 2.

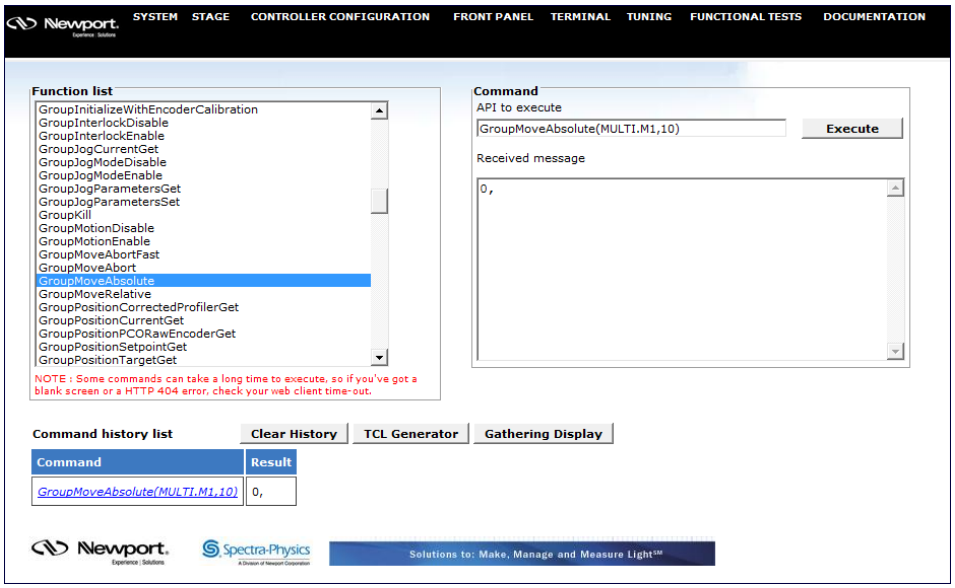


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3. When all arguments are defined, click “OK”. Now review the final syntax of the function and make final text changes, as needed. When done, click “Execute”.



4. When the function is executed, the controller’s response will appear in the Received message window. A returned 0 means that the function has been executed successfully. In all other cases, there will be an error code. Use the function ErrorStringGet() to get more information about the error.



The functions are listed in alphabetical order. Only those functions that are listed are available from the current system configuration. For example, if the system consists only of SingleAxis groups, no group specific functions for Spindles, XY groups, XYZ groups or MultipleAxis groups will be listed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

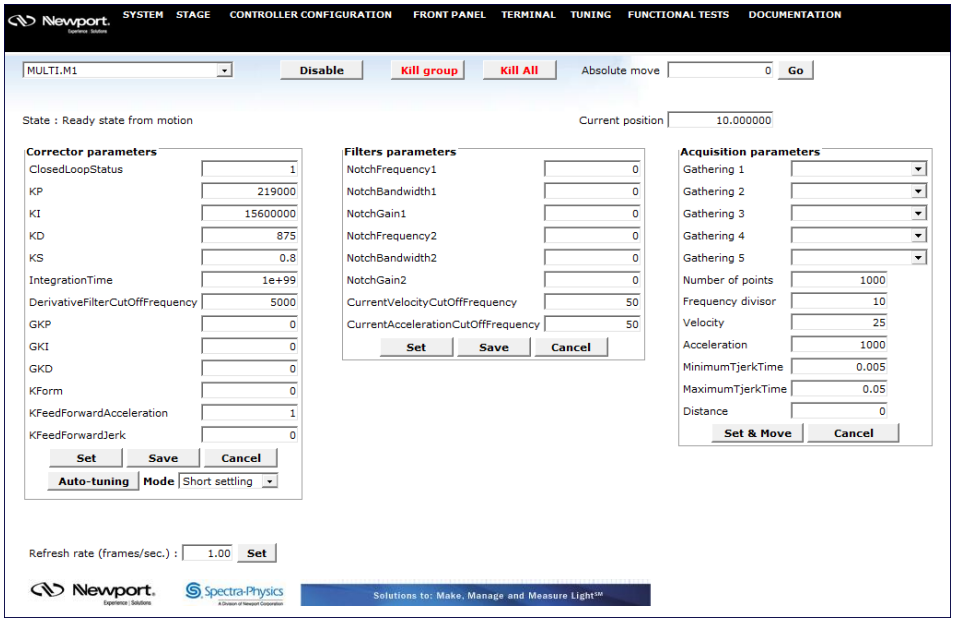
4.21 TUNING – Auto-Scaling

Auto-scaling is only available with positioners that feature a direct drive motor such as the XM, ILS-LM, IMS-LM or RGV100BL. To guarantee consistent performance of these stages, it is strongly recommended to perform Auto-scaling once the load is attached to the stage. During auto-scaling, the XPS controller measures the mass (inertia with rotation stages) on the positioner and returns recommended values for the Scaling Acceleration parameter.

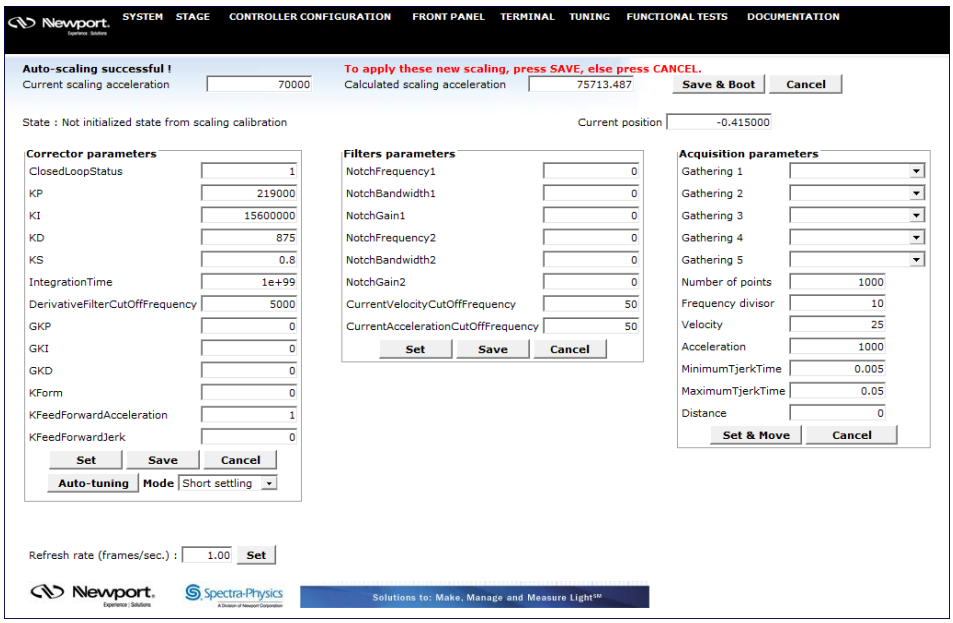
Repeat Auto-scaling with any major change of the payload on the positioner. With no major change of the payload, there is no need to redo Auto-scaling.

To perform Auto-scaling, do the following:

1. Select the main tab TUNING. Then select a positioner name. The following screen appears:



2. Click “Kill group”, then click “Auto-scaling”. The stage vibrates for a couple of seconds. Then, the following message appears:



3. To save the recommended values, click “Save”. To apply these new values, reboot the controller. The positioner should now work properly.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE

All other functions of the tuning page should be used only by experienced users.

4.22 TUNING – Auto-Tuning

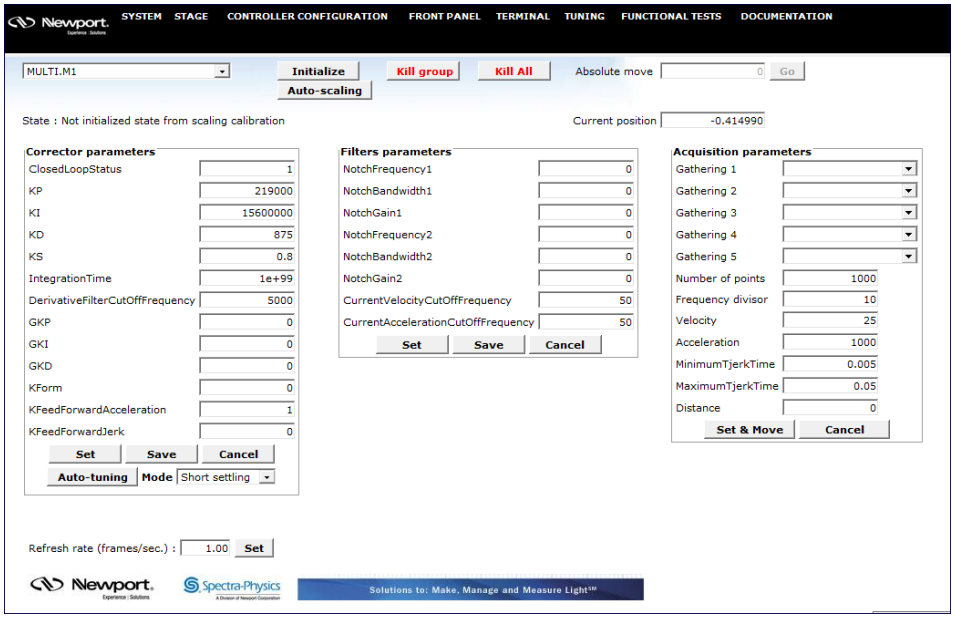
NOTE

Apart from the Auto-scaling feature, which is described in the previous chapter, only experienced motion control users should use the TUNING tool of the XPS controller.

All Newport positioners are supplied with default tuning parameters that provide consistently high performance for the vast majority of applications. Use the Tuning tool with Newport positioners only when not fully satisfied with the dynamic behavior of the positioners. Auto-Tuning works best with direct drive stages. Friction drive or ballscrew drive systems may not result in optimum tuning using this feature.

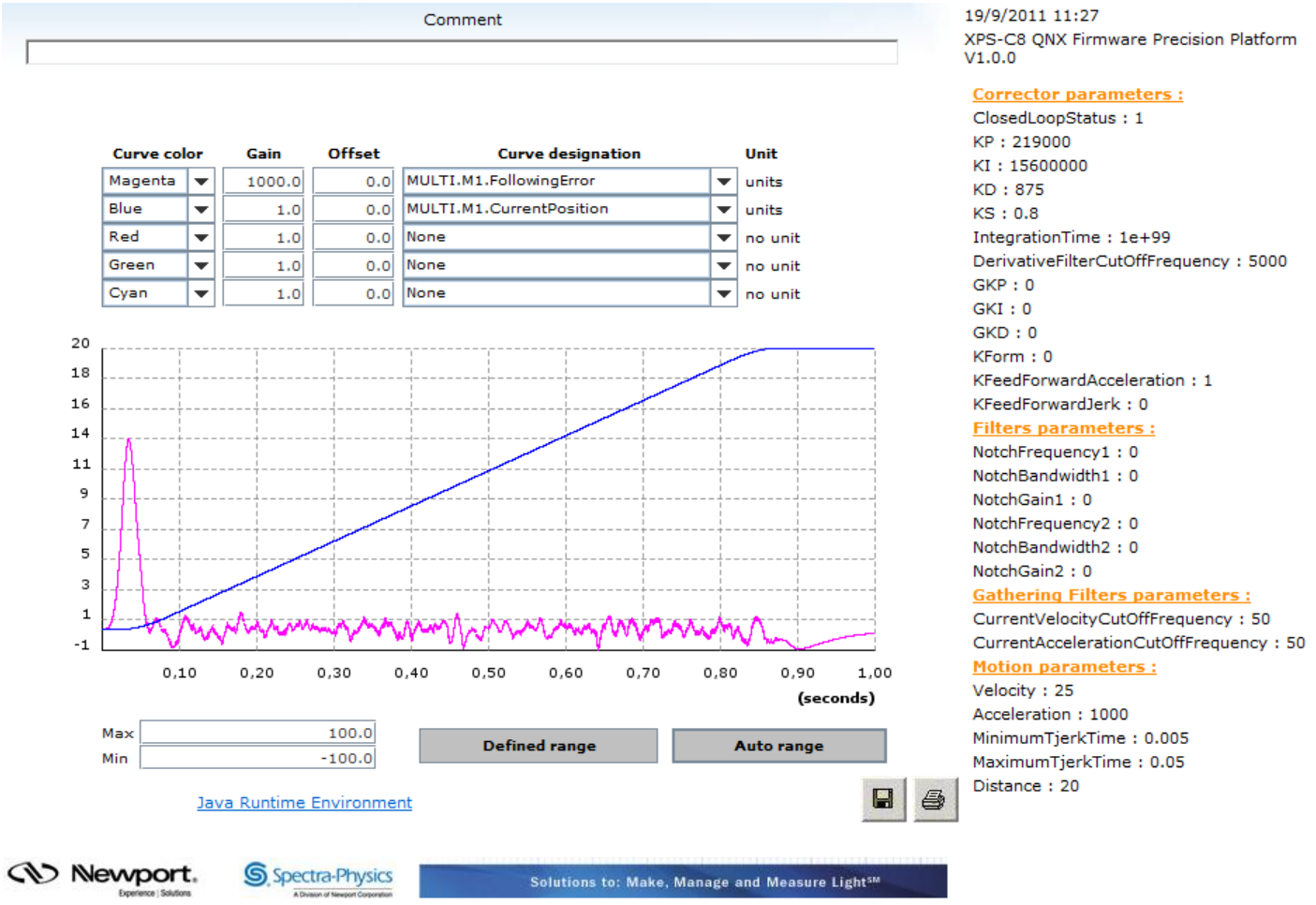
The following is a brief description of the TUNING tool:

1. Select a positioner name. The following screen appears:



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2. Perform a data gathering with your current parameter settings.
   1. Initialize and home the positioner, then move to the desired start position.
   2. Define the gathering data: For the stage tuning, it is recommended to gather only the following error and the current position.
   3. Define a typical motion distance.
   4. Define the frequency divisor. The frequency divisor defines the sampling rate of the gathering. A frequency divisor equal to one means one data point is gathered every servo cycle, or every 100 µs. With most positioners, it is sufficient to set a value of 10, meaning one data point every 1 ms.
   5. Define the number of points in relation to the distance, the frequency divisor, the velocity and the acceleration.
   6. Define the velocity, acceleration and jerk time.
   7. When done, click “Set & Move”.
3. The gathering results are displayed in a Java applet window. To view the results, install Java™ Runtime Environment Standard Edition on the host or remote computer. The XPS has a direct link to download Java™ Runtime when not installed on the computer.

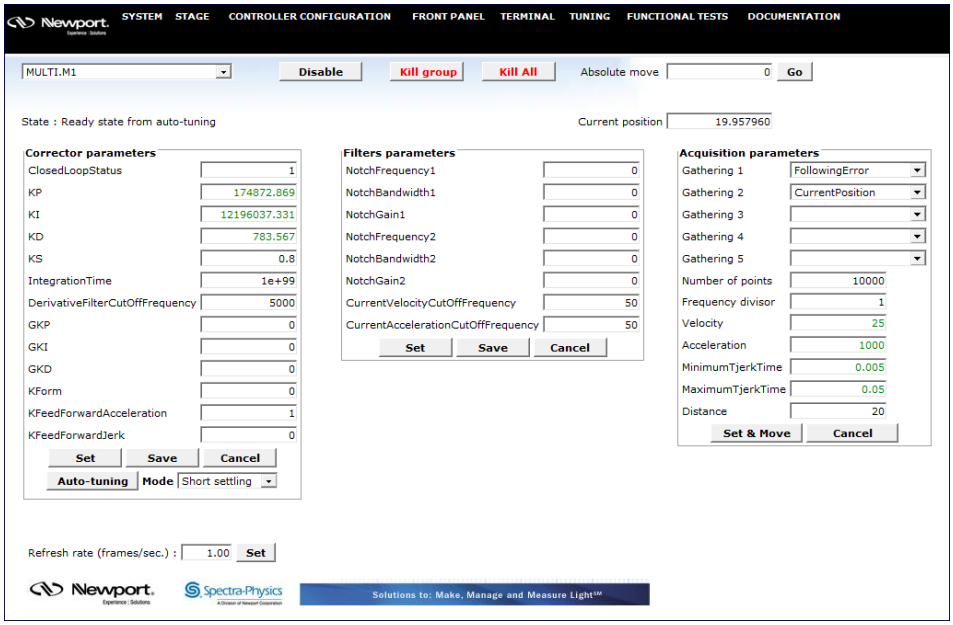


4. When satisfied with the results, there is no need to tune the stage. If not satisfied, return to the tuning page and move back to the start position.
5. Next to the Auto-tuning button, there is a Mode field for Auto-tuning. Select “Short settling” or “High robustness”. Choose “Short settling” to improve the settling time after a motion or to reduce the following error during the motion. Short settling will define “high” PID vales for your stage, but there is a risk of oscillation. Choose

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

"High robustness" to improve the robustness of the motion system and to avoid oscillations during or after a motion. "High robustness", for instance, can avoid oscillations for a rotation stage with high payload inertia. When done with the selection, click Auto-tuning.

6. The stage vibrates for a couple of seconds. When done the following screen appears:



7. Press "Set" to apply the new parameters. "Set" only changes the working parameters during data gathering. Recover the previous parameters by rebooting the system.
8. To test the behavior of the motion system with the new parameters, repeat the same data gathering and compare the results. Make manual changes to the settings and verify the behavior.
9. To permanently save the settings to the stages.ini, press "Save". "Save" overwrites the current settings in your stages.ini. Press "Save" only when fully satisfied with the results. For recovery, Newport recommends making a copy of the stages.ini with the old settings.

NOTE

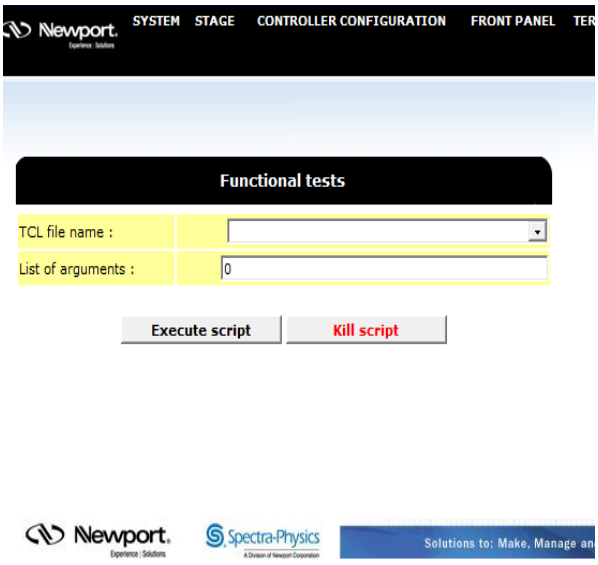
For further information about the meaning of the different tuning parameters, see chapter 14.0.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.23 FUNCTIONAL TESTS

The FUNCTIONAL TESTS page allows running TCL scripts saved in the "/Admin/Public/Scripts/ FunctionalTests" folder of the XPS controller. Supplied in the firmware, the Functional Tests scripts will then display the results of a gathering file.

Select the TCL Script name then press "Execute script" to run the script or "Kill script" to stop its execution.



4.24 FTP (File Transfer Protocol) Connection

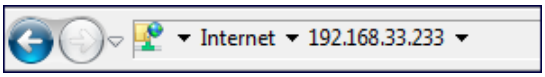
FTP is the protocol for exchanging files over the Internet. It works in the same way as HTTP for transferring web pages from a server to a user's browser and SMTP for transferring electronic mail across the Internet. FTP uses the Internet TCP/IP protocol to enable data transfer.

An FTP connection is needed to view the information saved in the XPS controller, to download documentation, to transfer configuration files (to modify them locally), to transfer TCL scripts, etc…

To connect to the FTP server:

- Start the XPS controller and wait until the boot sequence completes.
- Open an Internet browser window. Windows explorer is another option to access the files.
- Connect to the FTP server with the IP address of the controller:

Example

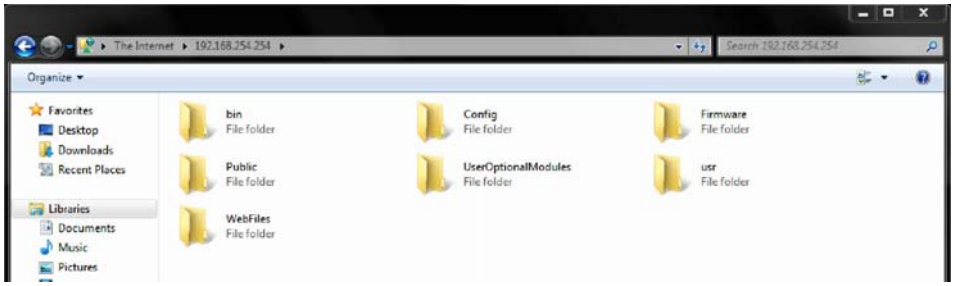


- Select "File" from the menu of the Internet browser, and then "Connect as…". The following window appears:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Specify the user name and password. Press log on. The folders of the XPS controller are displayed (see below). Browse through the different folders and transfer data from or to your host PC the same way as Windows Explorer.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.0 Maintenance and Service

5.1 Enclosure Cleaning

The XPS Controller/Driver should only be cleaned with a sufficient amount of soapy water solution. Do not use an acetone or alcohol solution, this will damage the finish of the enclosure.

5.2 Obtaining Service

The XPS Controller/Driver contains no user serviceable parts. To obtain information regarding factory service, contact Newport Corporation or your Newport representative and be ready with the following information:

- Instrument model number (on front panel) and original order number.
- Instrument serial number (on rear panel).
- Description of the problem.

If the XPS is to be returned to Newport Corporation, a Return Number will be issued, which should be referenced in the shipping documents.

Complete a copy of the Service Form found at the end of this User's Manual and include it with your shipment.

5.3 Troubleshooting

For troubleshooting, the user can query different error and status information from the controller. The XPS controller provides the Positioner Error, the Positioner Hardware Status, the Positioner Driver Status, the Group Status, and also a general system error.

If there is an error during command execution, the controller will return an error code. The command ErrorStringGet can be used to retrieve the description corresponding to the error code.

The following function commands are used to retrieve Positioner Error and Positioner Hardware Status:

- PositionerErrorGet: Returns an error code.
- PositionerErrorStringGet: Returns the description of the error code.
- PositionerHardwareStatusGet: Returns the status code.
- PositionerHardwareStatusStringGet: Returns the description corresponding to the status code.

In a fault condition, it is also very important to know the current status of the group and the cause of the transition from the previous group status to the current group state. The following functions can be used to retrieve the Group Status:

- GroupStatusGet: Returns the group status code.
- GroupStatusStringGet: Returns the description corresponding to the group status code.

NOTE

Refer to the Programmer's Manual for a complete list of status and error codes. Also refer to chapter 4.0 for troubleshooting the XPS controller with the help of its web utilities.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

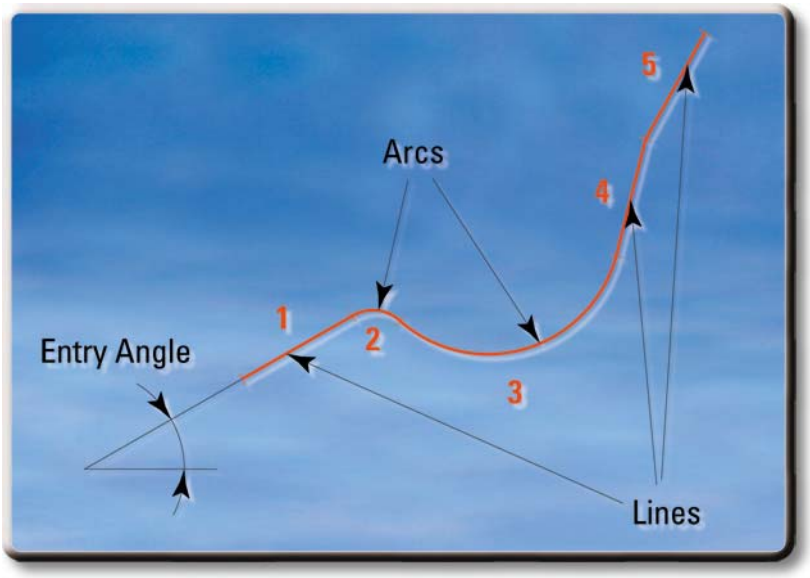
5.4 Updating the Firmware Version of Your XPS Controller

Users can regularly update the controller with new firmware releases. Updating the firmware does not overwrite the stages.ini or system.ini. No configuration will be lost when updating the firmware. Refer to the FirmwareHistory document which explains the changes needed in the stages.ini and system.ini files.

Refer to the XPS page at www.newport.com for more information.

A history file for the firmware and the stage database is added to the XPS web documentation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Motion Tutorial

6.0 XPS Architecture

6.1 Introduction

The architecture of the XPS firmware is based on an object-oriented approach. Objects are key to understanding this approach. Real-world objects share two characteristics: state and behavior. Software objects are modeled after real-world objects, so they have state and behavior too. A software object maintains its state in one or more variables. A variable is an item of data named by an identifier. A software object implements its behavior with methods. A method is a function (subroutine) associated with an object. Therefore, an object is a software bundle of variables and related methods. Encapsulating related variables and methods into a neat software bundle is a simple yet powerful idea that provides two primary benefits to software developers:

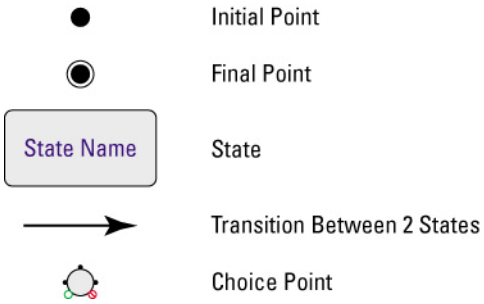
- **Modularity:** The source code for an object can be written and maintained independent of the source code for other objects. Also, an object can be easily passed around in the system.
- **Hidden information:** An object has a public interface that other objects can use to communicate with it. The object can maintain private information and methods that can be changed at any time without affecting the other objects that depend on it.

All objects have a life cycle and state diagrams are used to show the life cycle of the objects. The transition from one state to another is initiated after receiving a message from another object. Like all other diagrams, state diagrams can be nested in different layers to keep them simple and easy to read.

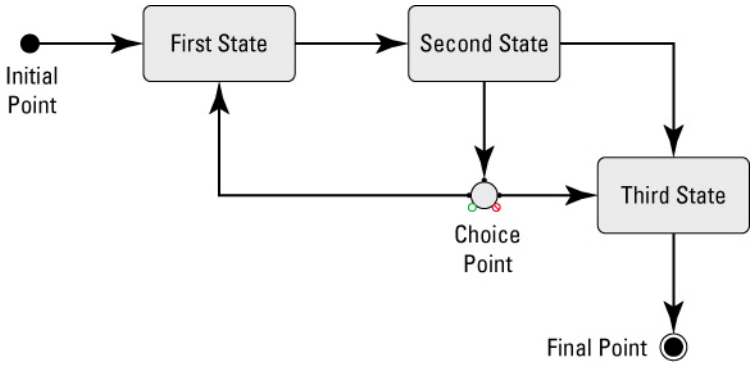
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6.2 State Diagrams

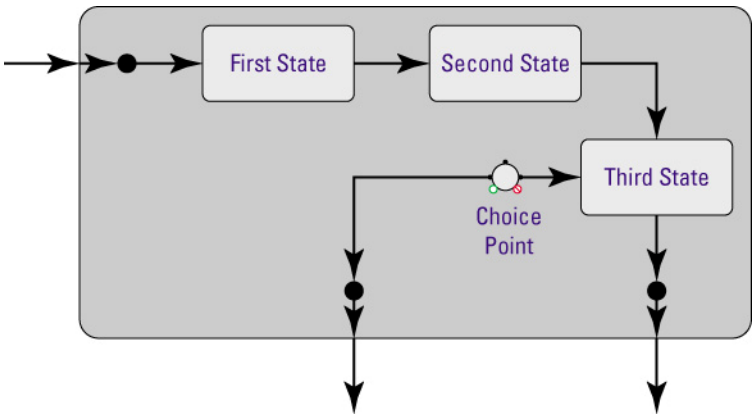
State diagrams are a way to describe the behavior of each group or object. They represent each steady state of a group and every transition between states in an exhaustive way. State diagrams contain the following components:



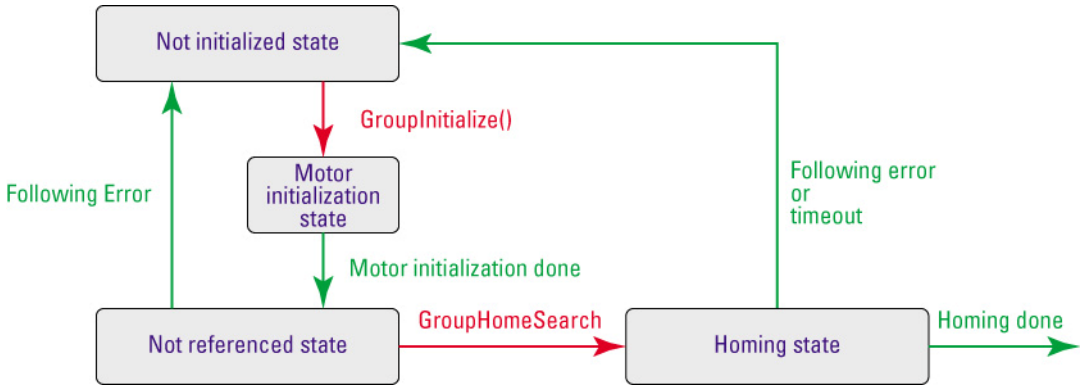
Here is an example of a simple stage diagram:



State diagrams can also include sub state diagrams:



The state diagrams that are specific to the XPS controller follow the same format. Within the XPS controller, all positioners are assigned to different motion groups. These motion groups have the following common state diagram:



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

As shown in the above state diagram, all groups have to be first initialized and then homed before any group is ready to perform any other function. Once the group is homed, it is in a ready state. There are five different motion groups available with the XPS controller:

- SingleAxis group
- Spindle group
- XY group
- XYZ group
- MultipleAxes group

Each group also has group specific states. Please refer to the Programmer's Manual for group-specific state diagrams for the five different groups.

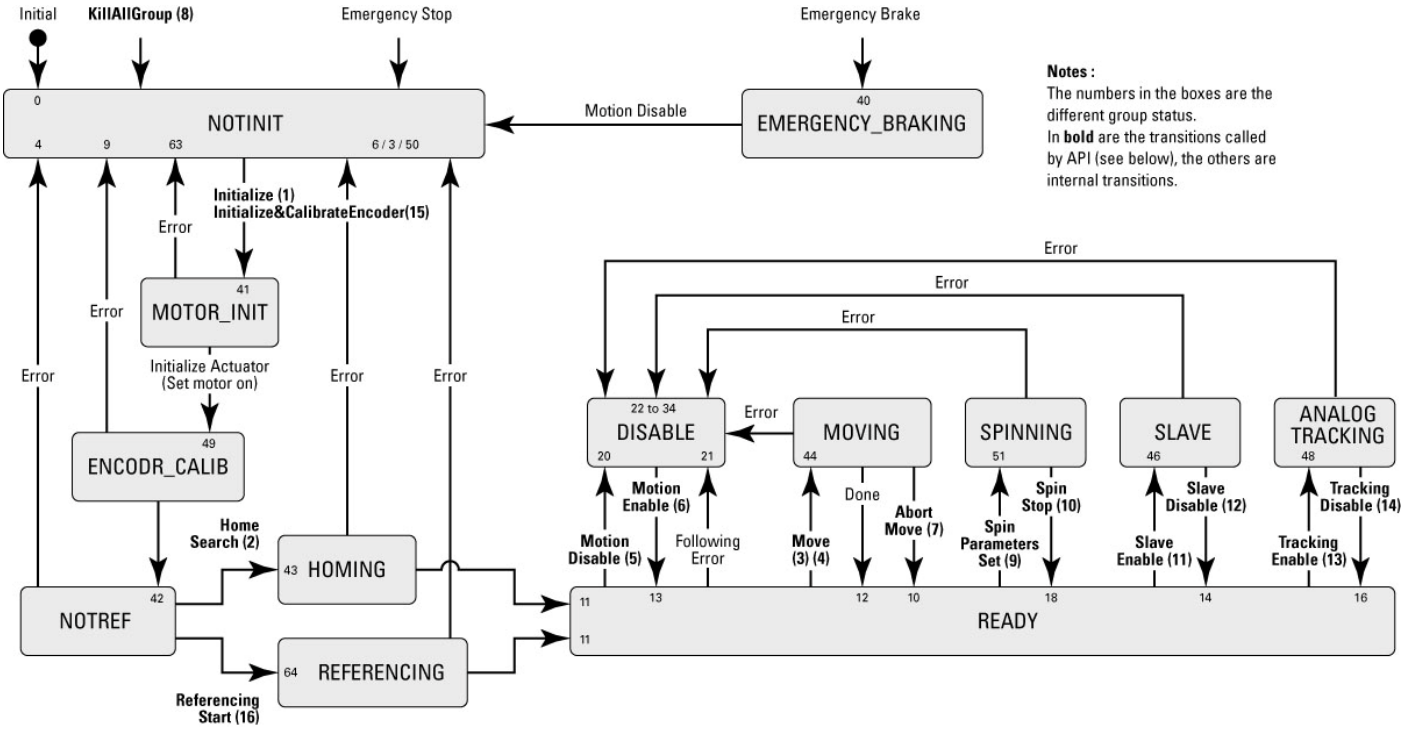
All positioners of a group are bundled together for security handling. Security handling of different groups is treated independently. Following is a list of the different faults and consequences that can happen in the XPS controller:

Error type	**Consequence**
General inhibition Motor fault Encoder fault	Emergency stop
End of travel	Emergency brake
Following error	Motion disable

- After an emergency brake or an emergency stop, both considered major faults, the corresponding group goes to a "not initialized" state: the system has to be initialized and homed again before any further motion.
- After a following error, as it is considered a minor fault, the corresponding group goes to a "Disable" state: a GroupMotionEnable() command puts the system back into "ready" state.

At any given time the group status can be queried from the controller. The function **GroupStatusGet (GroupName)** returns the current state number. The state numbers correspond to the state and to the event that generated the transition (if any). The function **GroupStatusStringGet (StateNumber)** returns the state description corresponding to the state number.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Called function

1. GroupInitialize	**7.** GroupMoveAbort	**13.** GroupAnalogTrackingModeEnable
2. GroupHomeSearch	**8.** GroupKill or KillAll	**14.** GroupAnalogTrackingModeDisable
3. GroupMoveAbsolute	**9.** GroupSpinParametersSet	**15.** GroupInitializeWithEncoderCalibration
4. GroupMoveRelative	**10.** GroupSpinModeStop	**16.** GroupReferencingStart
5. GroupMotionDisable	**11.** SpinSlaveModeEnable	**17.** GroupReferencingStop
6. GroupMotionEnable	**12.** SpinSlaveModeDisable	

State diagram of the XPS controller.

6.3 Motion Groups

Within the XPS controller, each positioner or axis of motion must be assigned to a motion group. This "group" can either be a SingleAxis group, a Spindle group, an XY group, an XYZ group or a MultipleAxes group. Once defined, the XPS automatically manages all safeties and trajectories of the motion group from the same function. For instance, the function **GroupHomeSearch (GroupName)** automatically homes the whole motion group GroupName independent of its definition as a SingleAxis group, a Spindle group, an XY group, an XYZ group or a MultipleAxes group. Within the system configuration file, system.ini, select the home sequence as "sequential", one positioner after the other, or in "parallel", with all positioners homing at the same time. With a single function such as **GroupMoveAbsolute (GroupName, Position),** the whole motion group, GroupName, is moved synchronously to the defined absolute position, where **"Position"** may be one or more parameters depending on the number of positioners this motion group contains. This same command can be used to move a single positioner of a group to an absolute position by using the syntax **GroupMoveAbsolute (GroupName.PositionerName, Position1)**. These powerful, object-oriented functions are not only extremely intuitive and easy to use, they are also more consistent with other programming methods and reduce the number of commands learned compared to traditional mnemonic commands.

Another benefit provided by motion groups is improved error handling. For instance, whenever an error occurs due to a following error or a loss of the end-of-run signal, only the motion group where the error originated is affected (disabled) while all other motion groups remain active and enabled. The XPS manages these events automatically. This greatly reduces complexity and improves the security and safety of sensitive applications.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To illustrate this, let's consider a typical scanning application. If there is an error on the stepping axis of the XY table (which is set-up as an XY group), only the XY table is disabled while the auto-focusing tool (a vertical stage that is defined as a separate SingleAxis group) continues to function.

Each of the five available motion groups has specific features:

6.3.1 Specific SingleAxis Group Features

Master-Slave – To enable this function, the slaved positioner must be defined as a SingleAxis group. The master positioner can be a member of any motion group. So it is possible to define a Positioner as a slave of another positioner that is part of an XYZ group.

6.3.2 Specific Spindle Group Features

The Spindle Group is a single positioner group that enables continuous rotations with no limits and with a periodic position reset.

Master-Slave - In Master-Slave spindle mode the master and the slave group must be Spindle groups.

6.3.3 Specific XY Group Features

Line-Arc trajectories, XY mapping – These features are only available with XY groups. It is not possible for an XY group to perform a Spline or a PVT trajectory. Also, an XY group cannot be slaved to another group, however, any positioner of an XY group can be a master to a slaved SingleAxis group.

6.3.4 Specific XYZ Group Features

Spline trajectories, XYZ mapping – These features are only available with XYZ groups. It is not possible for an XYZ group to perform a Line-Arc or a PVT trajectory. Also, an XYZ group cannot be slaved to another group, however, any positioner of an XYZ group can be a master to a slaved SingleAxis group.

6.3.5 Specific MultipleAxes Features

PVT trajectories – PVT trajectories are only available with MultipleAxes groups. It is not possible for a MutipleAxes group to perform a Line-Arc or a Spline trajectory. Also, a MultipleAxes group cannot be slaved to another group. However, any positioner of a MultipleAxes group can be a master to a slaved SingleAxis group.

6.4 Native Units

The XPS controller supports user-defined native units like µm, inches, degrees or arcsecs. The units for each positioner are set in the configuration file where the parameter EncoderResolution indicates the number of units per encoder count. When using the XPS controller with Newport stages, this part of the configuration is done automatically. Once defined, all motions, speeds and accelerations can be commanded in the same native unit without any math needed. All other parameters like stage travel, maximum speed and all compensations are defined on the same scale as well. This is a great advantage compared to other controllers that can be commanded only in multiples of encoder counts, which can be an odd number.

In the XPS controller there are 4 types of position information for each positioner: TargetPosition, SetpointPosition, FollowingError and CurrentPosition. These are described as follows:

The CurrentPosition is the current physical position of the positioner. It is equal to the encoder position after all compensations (backlash, linear error and mapping) have been taken into account.

The SetpointPosition is the theoretical position commanded to the servo loop. It is the position where the positioner should be, during and after the end of the move.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The FollowingError is the difference between the CurrentPosition and the SetpointPosition.

The TargetPosition is the position where the positioner must be after the completion of a move.

When the controller receives a new motion command after the previous move is completed, a new TargetPosition is calculated.

This new target is received as an argument for absolute moves. For relative moves, the argument is the length of the move and the new target is calculated as the addition of the current target and the move length. Then the profiler of the XPS calculates a set of SetpointPositions to determine where the positioner should be at each given time.

When the positioner is controlled by a digital servo loop with a PID corrector, part of the signals sent to the motor of the positioner is a function of the following error. Part of this function is the integral gain of the PID filter that requires a following error equal to zero to reach a constant value.

The encoder in the positioner delivers a discrete signal (encoder counts). Take the example of an encoder with a resolution of 1 and a target position equal to 1.4. The real position cannot reach the value of the target position (1 or 2 instead of 1.4), so the following error will never be equal to zero (closest values are +0.6 and -0.4). Thus, due to the integral gain of the PID filter, the system will never settle, but will oscillate between the positions 1 and 2.

The XPS controller avoids this instability while allowing the use of native units instead of encoder counts by using a rounded value of the TargetPosition to calculate the motion profile and a rounded value for the following error. But the non-rounded value of the TargetPostion will be stored as final position, so that there is no accumulation of errors due to rounding, in case of successive relative moves.

To understand the difference, consider a positioner with a resolution of 1 that is at the position 0. This positioner receives a relative motion command of 10.4. At the end of the motion the CurrentPosition will be 10 and the SetpointPosition will be 10, but the TargetPosition will be 10.4. The positioner then receives the same relative motion command again. At the end of this motion the CurrentPosition will be 21, the SetpointPosition will be 21 and the TargetPosition will be 20.8.

NOTE

When an application requires a sequence of small incremental motion of constant step size close to the encoder resolution, make sure that the commanded incremental motion is equal to a multiple of encoder steps.

The TargetPosition, SetpointPosition, CurrentPositon and FollowingError can be queried from the controller using the appropriate function calls.

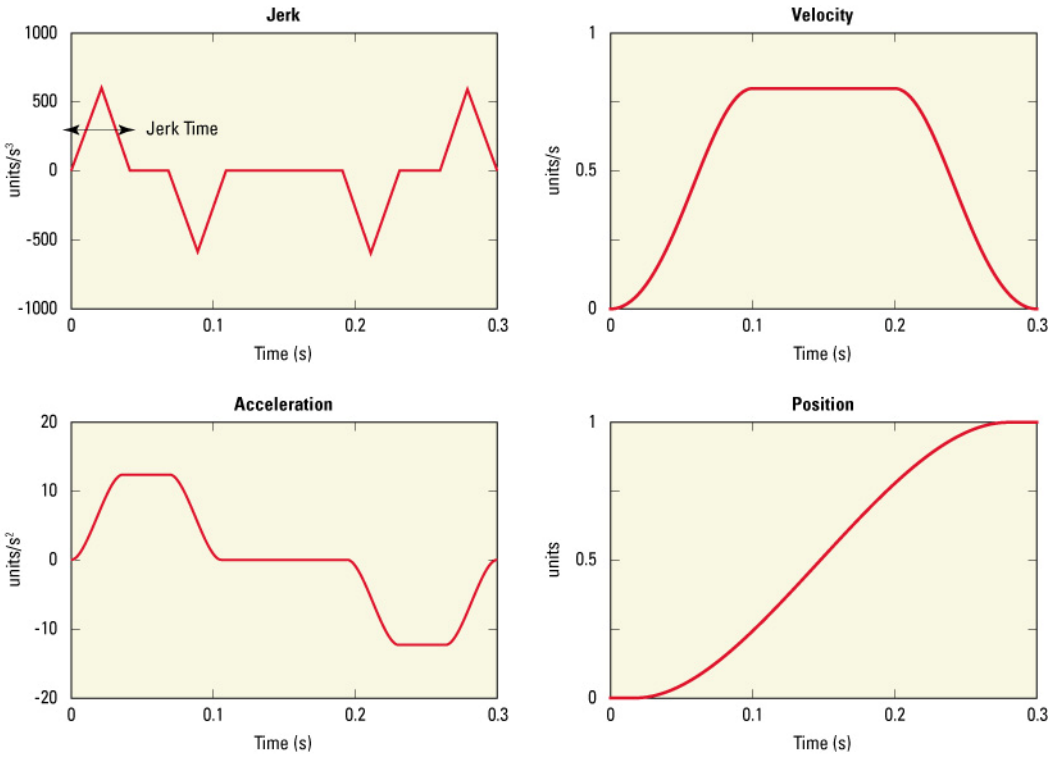
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

7.0 Motion

7.1 Motion Profiles

Motion commands refer to strings sent to a motion controller that will initiate a motion. The XPS controller provides several modes of positioning from simple point-to-point motion to the most complex trajectories. On execution of a motion command, the positioner moves from the current position to the desired destination. The exact trajectory for the motion is calculated by a motion profiler. So the motion profiler defines where each of the positioners should be at each point in time. There are details worth mentioning about the motion profiler in the XPS controller:

In a classical trapezoidal motion profiler (trapezoidal velocity profile), the acceleration is an abrupt change. This sudden change in acceleration can cause mechanical resonance in a dynamic system. In order to eliminate the high frequency portion of the excitation spectrum generated by a conventional trapezoidal velocity motion profile, the XPS controller uses a sophisticated SGamma motion profile. Figure 17 shows the acceleration, velocity and position plot for the SGamma profile.



Displacement: 150 e^{-3} units
Maximum velocity: 0.8 units/s
Maximum acceleration: 12 units/s^2
Minimum jerk time: 0.004 s
Maximum jerk time: 0.04 s

Notice: The minimum displacement lasts at least 4 times the minimum jerk time.

Figure 17: SGamma Motion Profile.

The SGamma motion profile provides better control of dynamic systems. It allows for perfect control of the excitation spectrum that a move generates. In a multi-axes system this profile gives better control of each axis independently, but also allows control of the cross-coupling that are induced by the combined motion of the axes. As shown in figure 17, the acceleration plot is parabolic. The parabola is controlled by the jerk time (jerk being the derivative of the acceleration). This parabolic characteristic of the acceleration

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

results in a much smoother motion. The jerk time defines the time needed to reach the necessary acceleration. One feature of the XPS controller is that it automatically adapts the jerk time to the step width by defining a minimum and a maximum jerk time. This auto-adaptation of the jerk time allows a perfect adjustment of the system's behavior with different motion step sizes.

NOTE

Because of jerk-controlled acceleration, any move has a duration of at least four times the jerk time.

For the XPS controller, the following parameters need to be configured for the SGamma profile:

- MaximumVelocity (units/s)
- MaximumAcceleration (units/s^2)
- EmergencyDecelerationMultiplier (Applies to Emergency Stop)
- MinimumJerkTime (s)
- MaximumJerkTime (s)

The above parameters are set in the stages.ini file for a positioner. When using the XPS controller with Newport stages, these parameters are automatically set during the configuration of the system.

The velocity, acceleration and jerk time parameters is modified by the function **PositionerSGammaParametersSet().**

Example

PositionerSGammaParametersSet (MyGroup.MyStage, 10, 80, 0.02, 0.02)

This function sets the positioner "MyStage" velocity to 10 units/s, acceleration to 80 units/s^2 and minimum and maximum jerk time to 0.02 seconds. The set velocity and acceleration must be less than the maximum values set in the stages.ini file. These parameters are not saved if the controller is shut down. After a re-boot of the controller, the parameters will retain the values set in the stages.ini file.

In actual use, the XPS places a priority on the displacement position value over the velocity value. To reach the exact position, the speed of the positioner may vary slightly from the value set in the stages.ini file or by the **PositionerSGammaParametersSet** function. So the drawback of the SGamma profile is that the velocity used during the move can be a little bit different from the velocity defined in the parameters. For example, the exact velocity will change when the move distance is changed, move 100mm, then 100.001 mm then 100.011 mm. There will be some changes to the commanded velocity. This change can be ignored for many applications except where an accurate time synchronization during the motion is required.

The function, **PositionerSGammaExactVelocityAdjustedDisplacementGet(),** can be used as described below to achieve the exact desired speed in applications that require an accurate value of the velocity during a move. In this case, the velocity value is adhered to, but the target position may be slightly different from the one required. In other words, according to the application requirements, the user can choose between very accurate positions or very accurate velocities.

Example

PositionerSGammaExactVelocityAdjustedDisplacementGet (MyGroup.MyStage, 50.55, ExactDisplacement)

This function returns the exact displacement for that move with the exact constant velocity set shown in the example above (10 mm/s). The result is stored in the variable ExactDisplacement, for instance 50.552.

GroupMoveAbsolute (MyGroup.MyStage, 50.552)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In the above example, for a position of 50.55 mm, the command returns a value of 50.552. This means that in order for the positioner "MyStage" to achieve the desired velocity in the most accurate way, the commanded position should be 50.552 mm instead of 50.55 mm.

The XPS can report two different positions. The first one is the SetpointPosition or theoretical position. This is the position where the stage should be according to the profile generator.

The second position is the CurrentPosition. This is the actual position as reported by the positioner's encoder after taking into account all compensation. The relationship between the SetpointPosition and the CurrentPosition is as follows:

Following error = SetpointPosition - CurrentPosition

The functions to query the SetpointPosition and the CurrentPosition values are:

GroupPositionCurrentGet() and GroupPositionSetpointGet()

7.2 Home Search

Home search is a specific motion process. Its goal is to define a reference point along the course of travel accurately and repeatably. The need for this absolute reference point is twofold. First, in many applications, it is important to know the exact position in space, even after a power-off cycle. Secondly, to prevent the motion device from hitting a travel obstruction set by the application (or its own hardware travel limits), the controller uses software limits. To be efficient, the software limits must be referenced accurately to the home before running the application.

After motor initialization, any motion group must first be homed or referenced before any further motion can be executed. Here, homing refers to a predefined motion process that moves a stage to a unique reference position and defines this as Home. Referencing refers to a group state that allows the execution of different motions and the setting of the position counters to any value (see next section for details). The referencing state provides flexibility for the definition of custom home search and system recovery processes. It should only be used by experienced users.

A number of hardware solutions may be used to determine the position of a motion device, the most common are incremental encoders. By definition, these encoders can only measure relative position changes and not absolute positions. The controller keeps track of position changes by incrementing or decrementing a dedicated counter according to the information received from the encoder. Since there is no absolute position information, position "zero" is where the controller was powered on (and the position counter was reset).

To determine an absolute position from incremental encoders, the controller must use a reference position that is unique to the entire travel, called a home switch or origin switch, usually in conjunction with an index pulse.

An important requirement is that this switch must have the same resolution as the encoder pulses.

If the motion device uses a linear scale as a position encoder, the home switch is usually placed on the same scale and read with the same resolution.

If, on the other hand, a rotary encoder is used, homing becomes more complicated. To have the same resolution, a mark on the encoder disk could be used (called index pulse), but because the mark repeats every revolution, it does not define a unique point over the entire travel. An origin switch, on the other hand, placed in the travel of the motion device is unique, but typically is not precise or repeatable enough. The solution is to use both in a dedicated search algorithm as follows.

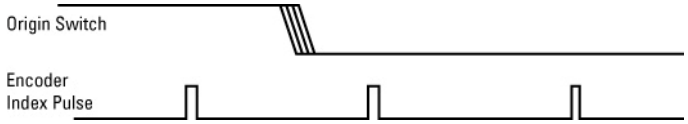


Figure 18: Home (Origin) Switch and Encoder Index Pulse.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A Home switch (Figure 18) separates the entire travel in two areas: one has a high level and the other has a low level. The most important part is the transition between the two areas. Just by looking at the origin switch level, the controller knows already on which side of the transition the positioner is and which direction to start the homing process.

The task of the home search process is to define one unique index pulse as the absolute position reference. This is first done by finding the home switch transition and then the very first index pulse (Figure 19).

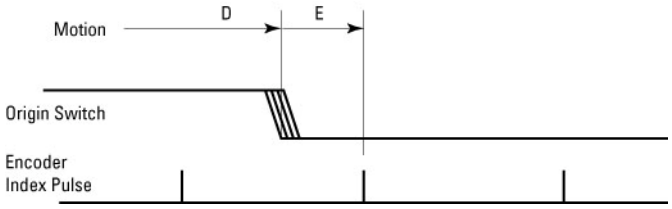


Figure 19: Slow-Speed Origin Switch Search.

Labeling the two motion segments D and E, the controller searches for the origin switch transition in D and for the index pulse in E. To guarantee the best repeatability possible, both D and E segments must perform at a very low speed and without stopping in between.

The homing process described above has a drawback. At low search speeds, the process could take a very long time if the positioner happens to start from the one end of travel. To speed things up, the positioner is moved fast until it is in the vicinity of the origin switch and then performs the two slow motions, D and E, at half the home search velocity. The new sequence is shown in Figure 20.

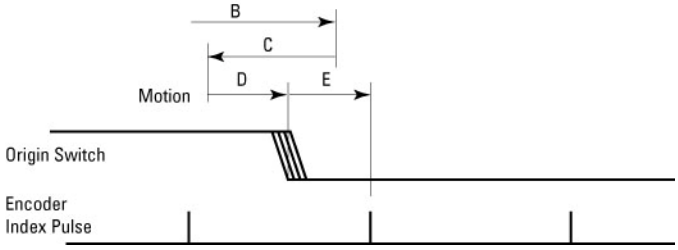


Figure 20: High/Low-Speed Home (Origin) Switch Search.

Motion segment B is performed at the pre-programmed home search speed. When the home switch transition is encountered, the motion device stops (with an overshoot), reverses direction and searches for the switch transition again, this time at half the speed (segment C). Once the switch transition is encountered, it stops again with an overshoot, reverses direction and executes D and E with one tenth of the programmed home search speed.

In the case when the positioner starts from the other end of the home switch transition, the routine is shown in Figure 21.

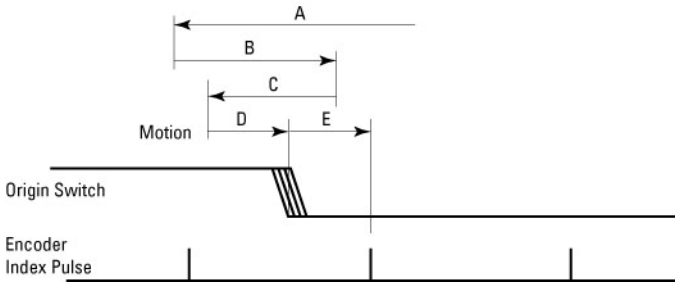


Figure 21: Home (Origin) Search from Opposite Direction.

The positioner moves at the home speed up to the home switch transition (segment A) and then executes segments B, C, D and E as in figure 20.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This home search process guarantees that the last segment, E, is always performed in the positive direction of travel and at the same reduced speed. This method ensures an precise and repeatable reference position.

There are 7 different home search processes available in the XPS controller:

1. **MechnicalZeroAndIndexHomeSearch** is used when the positioner has a hardware home switch plus a zero index from the encoder. This process is the default for most Newport standard stages.
2. **MechanicalZeroHomeSearch** is used with positioners that have a hardware home switch but with no zero index from the encoder.
3. **IndexHomeSearch** is used with positioners that have a home index, but with no hardware home switch signal. In this process, the positioner initially moves in the positive direction to find the index. When a + limit switch is detected, the direction of motion reverses until the index is found.

Note

For users with CIE03 (E3425x) boards, if a limit is detected before the index, there will be an emergency brake and the group will go in NOT_INITIALIZED status.

---

4. **CurrentPositionAsHome** is used when the positioner has no home switch or index. This process will keep the positioner's home at its current location. Setting the home too close to the EOR could generate unwanted emergency stops. Start with around 50 MIM (Minimum Incremental Movement) units, but an optimum distance may be determined by trial and error, depending on the stage.
   This feature can also be used to set home arbitrarily and bypass a home switch.
5. **MinusEndOfRunAndIndexHomeSearch** uses the positioner's minus end-of-run limit as a hardware home switch and a zero index from the encoder. This process is comparable to MechanicalZeroAndIndexHomeSearch, but uses the minus end-of-run limit signal as hardware home switch and moves in the positive direction until the Index is reached. Otherwise, it will reach the positive limit or a timeout will occur. The positioner homes to a position that is different from the MechanicalZeroAndIndexHomeSearch location.
6. **MinusEndOfRunHomeSearch** uses the positioner's minus end-of-run limit for homing. Note that the emergency stop at the negative limit is disabled during homing.
7. **PlusEndOfRunHomeSearch** uses the positioner's plus end-of-run limit for homing and the emergency stop at the positive limit is disabled during homing.

Note

This home search works only with the CIE05 (E3920x) board or later versions.

The home search process is set up in the stages.ini file. When using the XPS controller with Newport ESP-compatible stages, this setting is done automatically with the configuration of the system. The home search velocity, acceleration and time-out are also set up in the stages.ini file.

Each motion group can either be homed "together" or "sequentially", meaning all positioners belonging to that group home at the same time in parallel or all the positioners home one after the other, respectively. This option is also set up in the system.ini file or during configuration.

A Home search can be executed with all motion groups and any motion group MUST be homed before any further motion can be executed. To home a motion group that is in a "ready" state, that motion group must first be "killed" and then "re-initialized".

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

This is the sequence of functions that initialize and home a motion group.

GroupInitialize (MyGroup)

GroupHomeSearch (MyGroup)

…

GroupKill (MyGroup)

7.3 Referencing State

The predefined home search processes described in the previous section might not be compatible with all motion devices or might not be always executable. For instance, if there is a risk of collision during a standard home search process. In other situations, a home search process might not be desirable. For example, to ensure that the stages have not moved, the current positions are stored into memory. In this case, it is sufficient to reinitialize the system by setting the position counters to the stored position values.

For these special situations, the XPS controller's referencing state as in alternative to the predefined home search processes.

NOTE

The Referencing state should be only used by experienced users. Incorrect use could cause equipment damage.

The Referencing state is a parallel state to the homing state, see the state diagram on page 79, Figure 22. To enter the referencing state, send the function **GroupReferencingStart(GroupName)** while the group is in the NOT REFERENCED state.

In the Referencing state, the function **GroupReferencingActionExecute(PositionerName, Action, Sensor, Parameter)** will perform certain actions like moves, position latches of reference signal transitions, or position resets. The function **PositionerSGammaParametersSet(PositionerName)** can be used to change the velocity, acceleration and jerk time parameters.

To leave the referencing state, send the function **GroupReferencingStop(GroupName)**. The Group will then be in the HOMED state, state number 11.

The syntax and function of the function **GroupReferencingActionExecute(PositionerName, Action, Sensor, Parameter)** will be discussed in detail. With this function, there are four parameters to specify:

- PositionerName is the name of the positioner on which this function is executed.
- Action is the type of action that is executed. There are eight actions that can be distinguished into three categories: Moves that stop on a sensor event, moves of certain displacement, and position counter reset categories.
- Sensor is the sensor used for those actions that stop on a sensor event. It can be MechanicalZero, MinusEndOfRun, or None.
- Parameter is either a position or velocity value and provides further input to the function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following table summarizes all possible configurations:

Action	Sensor			Parameter	
	MechanicalZero	MinusEndOfRun	None	Position	Velocity
LatchOnLowToHighTransition	■	■			■
LatchOnHighToLowTransition	■	■			■
LatchOnIndex			■		■
LatchOnIndexAfterSensorHighToLowTransition	■	■			■
SetPosition			■	■	
SetPositionToHomePreset			■		
MoveToPreviouslyLatchedPosition			■		■
MoveRelative			■	■	

7.3.1 Move on sensor events

The “move on sensor events” starts a motion at a defined velocity, latches the position when a state transition of a certain sensor is detected, then stops the motion. There are four possible actions under this category:

- **LatchOnLowToHighTransition**
- **LatchOnHighToLowTransition**
- **LatchOnIndex**
- **LatchOnIndexAfterSensorHighToLow**

With **LatchOnLowToHighTransition** and **LatchOnHighToLowTransition**, latching happens when the right transition on the defined sensor occurs. The sensor can be latched to either **MechanicalZero**, **MinusEndOfRun** and **PositiveEndOfRun** when supported by the hardware, refer to §7.2 to know which hardware supports the function. With **LatchOnIndex** and **LatchOnIndexAfterSensorHighToLow**, latching happens on the index signal. With **LatchOnIndexAfterSensorHighToLow,** latching happens on the first index after a high to low transition at the defined sensor (**MechanicalZero** or **MinusEndOfRun**). Because of the dedicated hardware circuits used for the position latch, there is essentially no latency between sensor transition detection and position acquisition.

In all cases, motion stops after the latch. However, this means that the stopped position doesn’t rest on the sensor transition, but at some short distance from it. To move exactly to the position of the sensor transition, use the action **MoveToPreviouslyLatchedPosition**.

The latch does not change the current position value. In order to set the current position value, use the action **SetPosition** or **SetPositionToHomePreset**, for instance, after a **MoveToPreviouslyLatchedPosition**.

In the Referencing state, the limit switch safeties (emergency stop) are still enabled until the **MinusEndOfRun** sensor is specified with a **GroupReferencingActionExecute()** function. When specified, the limit switch safeties are disabled and will only be re-enabled with the function **GroupReferencingStop()**.

The Parameter has a sign, if it is assigned as velocity (floating point). This means that the direction of motion is dictated by the sign of the velocity parameter.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

7.3.2 Moves of Certain Displacements

These two move commands which don't use the same parameters, are explained below.

- **MoveRelative**

  The action **MoveRelative** commands a relative move of a positioner similar to the function **GroupMoveRelative**. However, the function **GroupMoveRelative** is not available in the Referencing state. The relative move is specified by a positive or negative displacement. The move is done with the SGamma profiler. The speed and acceleration are the default values, or the last value defined by either a move on sensor event, a **MoveToPreviouslyLatchedPosition**, or a **PositionerSGammaParametersSet**.

- **MoveToPreviouslyLatchedPosition**

  This action moves the positioner to the last latched position, see section 7.3.1: "Move on sensor events" for details. It verifies there was a position latched since this last **GroupReferencingStart** call. This is important because an old latched position can still be in memory from a previous home search or referencing. And moving to this previous latched position could have unexpected results. The move is done with the SGamma profiler. The speed is specified by a parameter. The acceleration is the default value, or the last value defined by a **PositionerSGammaParametersSet**.

7.3.3 Position Counter Resets

"Position counter resets" sets the current position to a certain value. There are two options: **SetPosition** and **SetPositionToHomePreset**. The main use of these actions is when the positioner is at a well defined reference position after a **MoveToPreviouslyLatchedPosition** action.

Another use of this action is for a "soft" system start by Referencing a group to a known set position, without executing a home search process, for example. In this case, a suggested sequence of functions follows:

GroupReferencingStart(GroupName)

GroupReferencingActionExecute(PositionerName, "SetPosition", "None", KnownCurrentPosition)

GroupReferencingStop(GroupName)

SetPosition sets the current position to a value defined by a parameter. **SetPositionToHomePreset** sets the current position to the **HomePreset** value stored in the stages.ini configuration file. It is equivalent to a **SetPosition** of the same positioner to the **HomePreset** value.

It is important that all positioners of a motion group are referenced to a position using the **SetPosition** or **SetPositionToHomePreset** before leaving the Referencing state (see example on page 94).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

7.3.4 State Diagram

The Referencing state is a parallel state to the homing state. It is between the NotReferenced state and the Ready state. Please see the state diagram below:

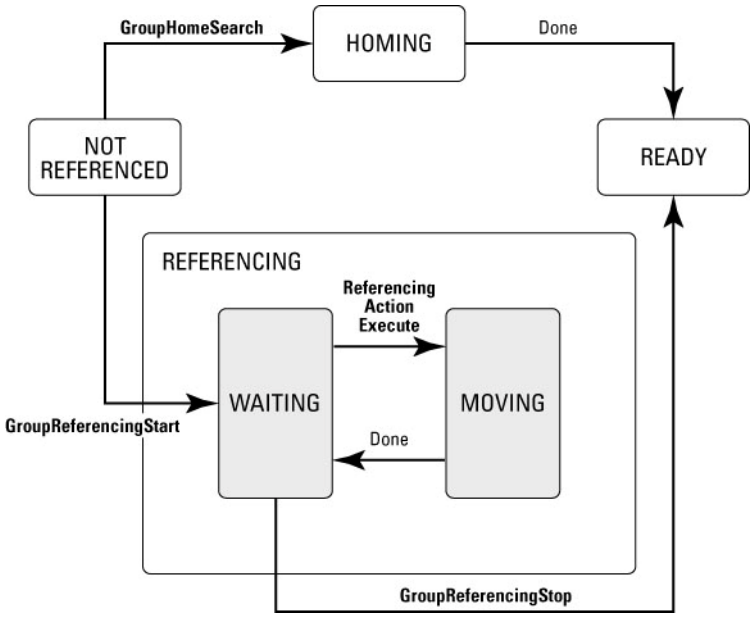


Figure 22: State Diagram.

7.3.5 Example: MechanicalZeroAndIndexHomeSearch

The following sequence of functions has the same effect as the MechanicalZeroAndIndexHomeSearch:

GroupReferencingStart(GroupName)
PositionerHardwareStatusGet (PositionerName, &status)
if ((status & 4) == 0) { // 4 is the Mechanical zero mask on the hardware status
GroupReferencingActionExecute(PositionerName, “LatchOnLowToHighTransition”, “MechanicalZero, -10) }
GroupReferencingActionExecute(PositionerName, “LatchOnHighToLowTransition”, “MechanicalZero”, 10)
GroupReferencingActionExecute(PositionerName, “LatchOnLowToHighTransition”, “MechanicalZero”, -5)
GroupReferencingActionExecute(PositionerName, “LatchOnIndexAfterSensorHighToLow”, “MechanicalZero”, 5)
GroupReferencingActionExecute(PositionerName, “MoveToPreviouslyLatchedPosition”, “None”, 5)
GroupReferencingActionExecute(PositionerName, “SetPositionToHomepreset”, “None”, 0)
GroupReferencingStop(GroupName)

7.4 Move

A move is a point-to-point motion. On execution of a move command, the motion device moves from a current position to a desired destination (absolute move) or by a defined increment (relative move). During motion, the controller is monitoring the feedback of the positioner and is updating the output based upon the following error. The XPS controller’s position servo is being updated at 10 kHz and the profile generator at 2.5 kHz, providing highly accurate closed loop positioning. Between the profiler and the corrector, there is a time-based linear interpolation to accommodate the different frequencies.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

There are two types of moves that can be commanded: an absolute move and a relative move. For an absolute move, the positioner will move relative to the HomePreset position as defined in the stages.ini file. In most cases the HomePreset is 0, which makes the home position equal to the zero position of the positioner. For a relative move, the positioner will move relative to the current TargetPosition. In relative moves, it is possible to make successive moves that are not equal to a multiple of an encoder step without accumulating errors.

Absolute and relative moves can be commanded to positioners and to motion groups. When commanding a move to a positioner, only the position parameter for that positioner must be provided. When commanding a move to a motion group, the appropriate number of position parameters must be provided with the move command. For instance for a move command to an XYZ group, 3 position parameters must be defined.

When commanding a move to a motion group, all positioners of that group will move synchronously. For any move, the controller will always determine the shortest time within the positioner's parameters setup. All positioners will start and stop their motion at the same time. This type of motion is also known as linear interpolation.

The functions for absolute and relative motions are **GroupMoveAbsolute()** and **GroupMoveRelative()** respectively.

Example

A motion system consisting of one XY group called ScanTable and one SingleAxis group called FocusStage. ScanTable has two positioners, called ScanAxis and StepAxis.

…

GroupHomeSearch (ScanTable)

GroupHomeSearch (FocusStage)

After homing is completed…

GroupPositionCurrentGet (ScanTable, Pos1, Pos2)

… will return 0 to Pos1 and 0 to Pos2, assuming PresetHome = 0.

GroupPositionCurrentGet (FocusStage, Pos3)

Will return 0 to Pos3, assuming HomePreset = 0.

GroupMoveAbsolute (ScanTable, 100, 50)

GroupMoveAbsolute (ScanTable.StepAxis, -20)

The second move is only for one positioner of that group and can be only executed after the first move is completed. After all moves are completed…

GroupPositionCurrentGet (ScanTable, Pos1, Pos2)

… will return 100 to Pos1 and -20 to Pos2.

GroupMoveRelative (FocusStage, 1)

GroupMoveRelative (FocusStage, 1)

The second move can be only executed after the first move is completed. After all moves are completed…

GroupPositionCurrentGet (FocusStage, Pos3)

… will return 2 to Pos3.

The velocity, acceleration and jerk time parameters of a move are defined by the function PositionerSGammaParametersSet() (see also section 7.1). When the controller receives new values for these parameters during the execution of a move, it will not take these new values into account on the current move, but only on the following moves. To change the velocity or acceleration of a positioner during the motion, use the Jogging mode (see section 7.5).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A move can be stopped at any time with the function **GroupMoveAbort()** that accepts GroupNames and PositionerNames. It is important to note, however, that the function **GroupMoveAbort(PositionerNames)** is accepted when the motion was commanded to the positioner, and not to the group. In the previous example, the function **GroupMoveAbort(ScanTable.ScanAxis)** is rejected for a motion that has been launched with **GroupMoveRelative(ScanTable, 100, 50)**. To stop this motion, send the function **GroupMoveAbort(ScanTable)**.

With XPS firmware 1.5.0 and higher, the XPS controller supports also asynchronous moves of several positioners belonging to the same motion group. The individual motion, however, needs to be managed by separate threads (see also section 18.4 for details).

7.5 Motion Done

The XPS controller supports two methods that define when a motion is completed (MotionDone): the theoretical MotionDone and the VelocityAndPositionWindow MotionDone. The method used is set in the stages.ini file. In theory, MotionDone is completed as defined by the profiler. However, it does not take into account the settling of the positioner at the end of the move. So depending on the precision and stability requirements at the end of the move, the theoretical MotionDone might not always be the same as the physical end of the motion. The VelocityAndPositionWindow MotionDone allows a more precise definition by specifying the end of the move with a number of parameters that take the settling of the positioner into account. In the VelocityAndPositionWindow MotionDone, the motion is completed when:

> | PositionErrorMeanValue | < | MotionDonePositionThreshold | AND | VelocityMeanValue | < | MotionDoneVelocityThreshold | is verified during the MotionDoneCheckingTime period.

The different parameters have the following meaning:

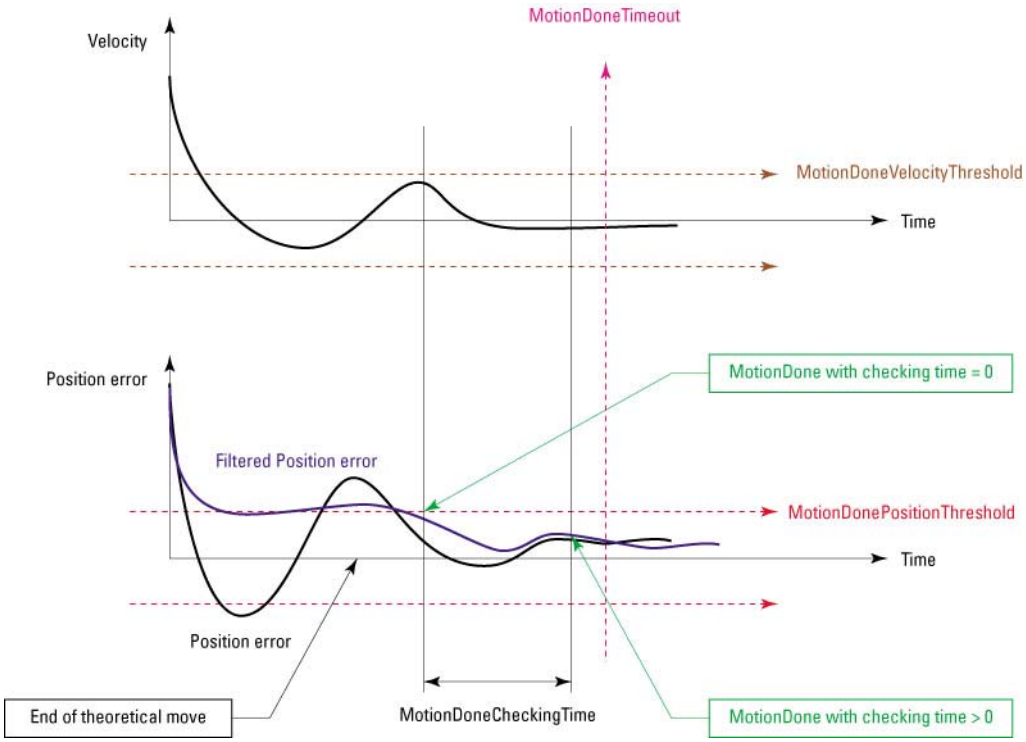


Figure 23: Motion Done.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **MotionDonePositionThreshold**: This parameter defines the position error window. The position error has to be within ± of this value for a period of MotionDoneCheckingTime to validate this condition.
- **MotionDoneVelocityThreshold**: This parameter defines the velocity window. The velocity at the end of the motion has to be within ± of this value for a period of MotionDoneCheckingTime to validate this condition.
- **MotionDoneCheckingTime**: This parameter defines the period during which the conditions for the MotionDonePositionThreshold and the MotionDoneVelocityThreshold must be true before setting the motion done.
- **MotionDoneMeanPeriod**: A sliding mean filter is used to attenuate the noise for the position and velocity parameters. The MotionDoneMeanPeriod defines the duration for calculating the sliding mean position and velocity. The mean position and velocity values are compared to the threshold values as defined above. This parameter is not illustrated on the graph.
- **MotionDoneTimeout**: This parameter defines the maximum time the controller will wait from the end of the theoretical move for the MotionDone condition, before sending a MotionDone time-out.

Important:

The XPS controller can only execute a new move on the same positioner or on the same motion group when the previous move is completed (MotionDone) and when the positioner or the motion group is again in the ready state.

The XPS controller allows triggering an action when the motion is completed (MotionDone) by using the event MotionEnd. For further details see chapter 11.0.

The functions **PositionerMotionDoneGet()** and **PositionerMotionDoneSet()** allow reading and modifying the parameters for the VelocityAndPositionWindow MotionDone. These parameters are only taken into account when the MotionDoneMode is set to VelocityAndPositionWindow in the stages.ini.

Example

Modifications of the MotionDoneMode can be made only manually in the stages.ini file. The stages.ini file is located in the config folder of the XPS controller, see Chapter 5 “FTP connection“ for details. Stage parameters can also be modified from the website, in Administrator mode, STAGES menu, Modify submenu.

Make a copy of the stages.ini file to the PC. Open the file with any text editor and modify the MotionDoneMode parameter of the appropriate stage to VelocityAndPositionWindow, and set the following parameters:

```
;--- Motion done
MotionDoneMode = VelocityAndPositionWindow  ; instead of Theoretical
MotionDonePositionThreshold = 4             ; units
MotionDoneVelocityThreshold = 100           ; units/s
MotionDoneCheckingTime = 0.1                ; seconds
MotionDoneMeanPeriod = 0.001                ; seconds
MotionDoneTimeout = 0.5                     ; seconds
```

Replace the current stages.ini file on the XPS controller with this modified version (make a copy of the old .ini file first). Reboot the controller. To apply any changes to the stages.ini or system.ini, the controller has to reboot.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Use the following functions:

GroupInitialize(MyGroup)

GroupHomeSearch(MyGroup)

PositionerMotionDoneGet(MyGroup.MyPositioner)

This function returns the parameters for the VelocityAndPositionWindow Motion done previously set in the stages.ini file, so 4, 100, 0.1, 0.001 and 0.5.

PositionerMotionDoneSet(MyGroup.MyPositioner, PositionThresholdNewValue, VelocityThresholdNewValue, CheckingTimeNewValue, MeanPeriodNewValue, TimeoutNewValue)

This function replaces the parameters with the newly entered values. If this function is not executed, the default setting from the .ini file is used.

7.6 JOG

Jog is an indeterminate motion defined by velocity and acceleration. Unlike a **GroupMoveAbsolute()** or a **GroupMoveRelative()**, the end of the motion is not defined by a target position. It can be best described by a "go"-command with a definition how fast, but not how far.

In Jog mode, the speed and acceleration of a motion group can be changed on-the-fly to accommodate varying situations. This is not possible with a **GroupMoveAbsolute()** or a **GroupMoveRelative()** which are defined moves. Practical examples for Jog are with tracking systems or coordinate transformations where the speed or acceleration of the jogging group is modified depending on the position or speed of the other motion groups or based on an analog input value.

The Jog mode can be enabled using the function **GroupJogModeEnable()** and is available to all motion groups. Once this mode is enabled, the motion parameters can be set using the command **GroupJogParameterSet()** which is applicable to positioners and to motion groups. To exit the Jog mode, first set the velocity to zero and then send the function **GroupJogModeDisable().**

Examples

For a single axis group:

GroupJogModeEnable (MySingleGroup)

Enables the Jog mode.

GroupJogParameterSet (MySingleGroup, 5, 20)

The single stage starts moving with a velocity of 5 units per second and an acceleration of 20 units per second^2.

GroupJogParameterSet (MySingleGroup, -5, 20)

The single stage starts moving in the reverse direction with the same velocity and same acceleration.

GroupJogParameterSet (MySingleGroup, 0, 20)

The single stage stops moving, its velocity being 0 units per second.

GroupJogModeDisable (MySingleGroup)

Disables the Jog mode.

For an XY group:

GroupJogModeEnable (MyXYGroup)

Enables the Jog mode.

GroupJogParameterSet (MyXYGroup, 5, 20, 10, 40)

The X axis and Y axis start moving with a velocity of 5 and 10 units per second and an acceleration of 20 and 40 units per second^2 respectively.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GroupJogParameterSet (MyXYGroup, 0, 20, 0, 40)

Both stages stop moving, their velocities being 0 units per second.

To apply new parameters to only one stage, use the following function:

GroupJogParameterSet (MyXYGroup.XPositioner, 5, 20)

Only the X axis starts moving with a velocity of 5 units per second and an acceleration of 20 units per second^2.

GroupJogParameterSet (MyXYGroup.XPositioner, 0, 20)

The X axis stage stops moving, its velocity being 0 units per second.

GroupJogModeDisable (MyXYGroup)

Disables the Jog mode.

In Jog mode, the profiler uses the CurrentPosition and the defined velocity and acceleration to calculate a new Setpoint position every 0.4 ms. These new Setpoint positions are then transferred to the corrector loop which runs every 0.1 ms. To accommodate the different frequencies between the profiler and the corrector, a linear interpolation between the new Setpoint and the previous Setpoint is done. Worst case, a new velocity and acceleration can be executed only every 0.4 ms. In Jog mode, the profiler uses a trapezoidal motion profile (see also section 7.1 for further details on motion profiles).

7.7 Master Slave

In master slave mode, any motion axis can be electronically geared to another motion axes, or a single master with multiple slaves. The gear ratio between the master and the slave is user defined. During motion, all axes compensations of the master and the slave are taken into account.

The slave must be a SingleAxis group. The master can be a positioner from any group. The Master slave relation is set by the function **SingleAxisSlaveParametersSet().**

The Master slave mode is enabled by the function **SingleAxisSlaveModeEnable()**. To enable the Master slave mode, the Slave group must be in the ready state. The Master group can be in the not-referenced or ready state.

Example 1

This example shows the sequence of functions used to set-up a master-slave relation between two axes that are not mechanically joined (meaning the two axis can move independently):

GroupInitialize (SlaveGroup)

GroupHomeSearch (SlaveGroup)

GroupInitialize (MasterGroup)

GroupHomeSearch (MasterGroup)

…

SingleAxisSlaveParametersSet (SlaveGroup, MasterGroup.Positioner, Ratio)

SingleAxisSlaveModeEnable (SlaveGroup)

GroupMoveRelative (MasterGroup.Positioner, Displacement)

…

SingleAxisSlaveModeDisable (SlaveGroup)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example 2

This example shows the sequence of functions used to set-up a Master slave relation **between two axes that are mechanically joined**. Different from example 1, all motions, including the motion done during the home search routine, are performed synchronously.

Important: First, set the HomeSearchSequenceType of the Slave group's positioner to CurrentPositionAsHome in the stages.ini and reboot the XPS controller.

GroupInitialize (SlaveGroup)

GroupHomeSearch (SlaveGroup)

GroupInitialize (MasterGroup)

SingleAxisSlaveParametersSet (SlaveGroup, MasterGroup.Positioner, Ratio)

SingleAxisSlaveModeEnable (SlaveGroup)

GroupHomeSearch (MasterGroup)

…

GroupMoveRelative (MasterGroup.Positioner, Displacement)

NOTE

The slave positioners should have similar capabilities as the master positioner in terms of velocity and acceleration. Otherwise the full capabilities of the master or the slave positioners may not be utilized.

7.8 Analog Tracking

Analog tracking controls the position or velocity of a motion group via external analog inputs. Analog tracking is available with all motion groups. To enable this mode, first set the tracking parameters of the positioners belonging to that motion group. Then enable tracking while the motion group is homed (in ready state after homing). In analog tracking mode, the analog inputs are filtered by a first order low-pass filter. Its cut-off frequency is defined by the parameter "TrackingCutOffFrequency" given in the section "profiler" of the stage.ini parameter file.

To set or get the tracking parameters, use the following functions:

PositionerAnalogTrackingPositionParametersSet()

PositionerAnalogTrackingPositionParametersGet()

…

PositionerAnalogTrackingVelocityParametersSet()

PositionerAnalogTrackingVelocityParametersGet()

The functions PositionerAnalogTrackingPositionParametersSet() and PositionerAnalogTrackingVelocityParametersSet() define the maximum velocity and acceleration used during analog tracking.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

7.8.1 Analog Position Tracking

The parameters that can be set for analog position tracking are the GPIO Name, scale and offset. The GPIO Name denotes which connector and pin number the analog signal will be input. The scale and the offset are used to calibrate the output position in the following way:

Position = InitialPosition + (AnalogValue - Offset) * Scale

Typical applications of analog position tracking are for beam stabilization, tracking systems, auto focusing sensors or alignment systems. When connecting a function generator to the GPIO input, analog tracking provides an easy way to make cyclical or sinusoidal motion, for example.

Example

Following is an example that shows the sequence of functions used to setup Analog Position Tracking:

GroupInitialize (Group)

GroupHomeSearch (Group)

…

PositionerAnalogTrackingPositionParameterSet (Group.Positioner, GPIO2.ADC1, Offset, Scale, Velocity, Acceleration)

GroupAnalogTrackingModeEnable (Group, "Position")

…

GroupAnalogTrackingModeDisable (Group)

7.8.2 Analog Velocity Tracking

The parameters that can be set for analog velocity tracking are the GPIO Name, offset, scale, deadband threshold and order. The relationship among offset, scale, deadband and order is illustrated in Figure 24.

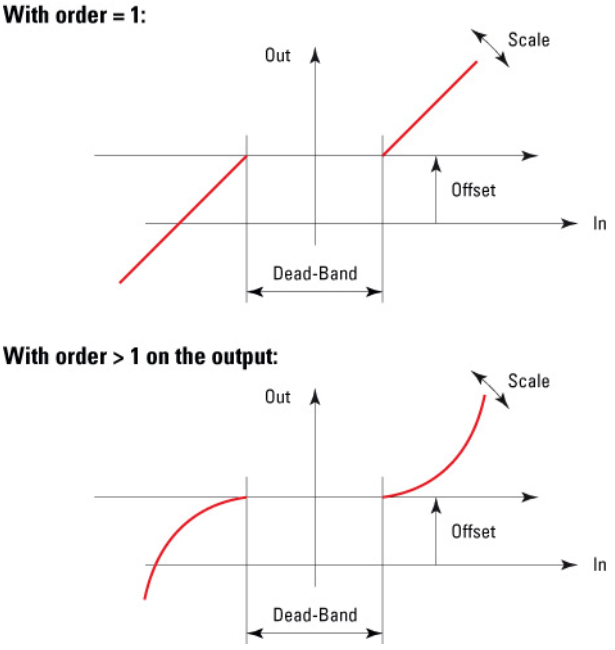


Figure 24: The Relationship Among Offset, Scale, Dead Band & Order.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The tracking velocity calculates as follows:

- AnalogInput is the voltage input at the GPIO
- AnalogGain refers to the AnalogGain setting of the analog input
- Offset, Order, DeadBandThreshold, and scale are defined with the function PositionerAnalogTrackingVelocityParametersSet
- MaxADCAmplitude, InputValue, OutputValue are internally-used parameters only

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|/ MaxADCAmplitude) * Order

Velocity = Sign(InputValue) * OutputValue * Scale * MaxADCAmplitude

In the dead band region there is no motion. If the order is set to 1, then the velocity is linear with respect to the input voltage.

If order is set greater than 1, then the velocity response is polynomial with respect to the input voltage. This makes the change in velocity more gradual and more sensitive in relation to the change in voltage.

A good example for using analog velocity tracking is for an analog joystick.

Example

Following is an example that shows the sequence of functions used to set-up Analog Velocity Tracking:

GroupInitialize (Group)

GroupHomeSearch (Group)

…

PositionerAnalogTrackingVelocityParameterSet (Group.Positioner, GPIO2.ADC1, Offset, Scale, DeadBandThreshold, Order, Velocity, Acceleration)

GroupAnalogTrackingModeEnable (Group, "Velocity")

…

GroupAnalogTrackingModeDisable (Group)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8.0 Trajectories

The XPS controller supports 3 different types of trajectories:

The Line-arc trajectory is a trajectory defined by a combination of straight and curved segments. It is available only for positioners in XY groups. The major benefit of a Line-arc trajectory is the ability to maintain constant speed (speed being the scalar of the trajectory velocity) throughout the entire path, excluding the acceleration and deceleration periods. The trajectory is user defined in a text file that is sent to the controller via FTP. Once defined, the user executes a function to begin the trajectory and the XPS automatically calculates and executes the motion, including precise monitoring of the speed and acceleration all along the trajectory. Simply executing the same trajectory more than once results in continuous path contouring. A dedicated function performs a precheck of the trajectory which returns the maximum and minimum travel requirements per positioner as well as the maximum possible trajectory speed and trajectory acceleration that is compatible with the different positioner parameters.

The spline trajectory executes a Catmull-Rom spline (which is a 3rd order polynomial curve) on an XYZ group. The main requirements of a spline are to hit all points (except for the first and the last point that are only needed to define the start and the end of the trajectory) and to maintain a constant speed throughout the entire path (except during the acceleration and deceleration period). The definition and execution of the spline trajectory is similar to the Line-arc trajectory with similar functions for trajectory pre-checking.

The PVT-mode is the most complex trajectory and is only available with MultipleAxes groups. In a PVT trajectory, each trajectory element is defined by the end position and end speed of each positioner plus the move time for the element. When all elements are defined, the controller calculates the cubic function trajectory that will pass through all defined positions at the defined times and velocities. PVT is a powerful tool for any kind of trajectory with varying speeds and for trajectories with rotation stages or other nonlinear motion devices.

8.1 Line-Arc Trajectories

8.1.1 Trajectory Terminology

Trajectory: defined as a continuous multidimensional motion path. Line-arc trajectories are defined in a two-dimensional XY plane. These are used with XY groups. The main requirement of a Line-arc trajectory is to maintain a constant speed (speed being the scalar of the vector velocity) throughout the entire path (except during the acceleration and deceleration periods).

Trajectory element (segment): an element of a trajectory is defined by a simple geometric shape, in this case a line or an arc segment.

Trajectory velocity: the tangential linear velocity (speed) along the trajectory during its execution.

Trajectory acceleration: the tangential linear acceleration used to start and end a trajectory. Trajectory acceleration and trajectory deceleration are equal by default.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8.1.2 Trajectory Conventions

When defining and executing a Line-arc trajectory, a number of rules must be followed:

- The motion group must be an XY group.
- All trajectories must be stored in the controller's memory under ..\public\trajectories (one file for each trajectory). Once a trajectory is started, it executes in the background allowing other groups or positioners to work independently and simultaneously.
- Each trajectory must have a defined beginning and end. Endless (infinite) trajectories are not allowed. Although, N-times (N defined by user) non-stop execution of the same trajectory is allowed. As the trajectory is stored in a file, the trajectory's maximum size (maximum elements number) is unlimited for practical purposes.
- Two types of Line-arc trajectory elements (segments) are available: lines Line(X,Y) and arcs Arc(R,A) (Radius, SweepAngle). Any Line-arc trajectory is a set of consecutive line or arc segments. The line segments are true linear interpolations $y = A*x + B$, the arc segments are true arcs of circles $(x - x0)^2 + (y - y0)^2 = R^2$.
- A Line-arc trajectory forms a continuous path, so each segment's final position is equal to the next segment's starting position. However, as the segment's tangential angles around the connection point of any two consecutive segments may not be continuous, there might be velocity discontinuities from one segment to next. For reference, this discontinuity is categorized as R0, wherein the position is continuous, but velocity is not. An excessive velocity discontinuity at joints can damage the stages, so the trajectory definition process must take this into account.
- Each Line-arc trajectory element is defined relative to the trajectory starting point. Every trajectory starting point has the coordinates (0,0), which has no relation to the zero position of the positioners. All trajectories physically start from the current X and Y positions of the XY group.

8.1.3 Geometric Conventions

The coordinate system of a Line-arc trajectory is an XY orthogonal system.

The X-axis of this system correlates to the XPositioner and the Y-axis correlates to the YPositioner of the XY group as defined in the system.ini.

The origin of the XY coordinate system is in the lower left corner, with positive values up and to the right.

All angles are measured in degrees, presented as floating point numbers. Angle origin and signs follow the trigonometric convention: positive angles are measured counter-clockwise.

8.1.4 Defining Line-Arc Trajectory Elements

A Line-arc trajectory is defined by a number of line and arc elements. The trajectory elements are executed in the same order as defined in the trajectory data file.

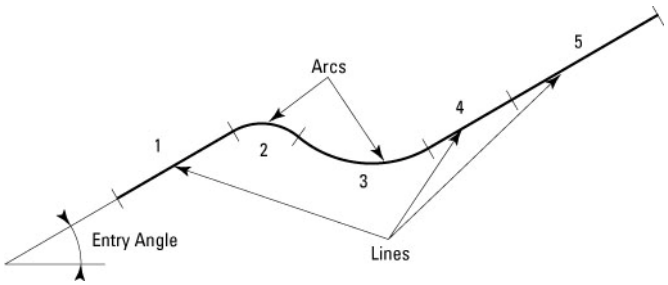


Figure 25: Line-arc trajectory example.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Figure 25 shows a trajectory example. Every trajectory must have a first element entry angle (called First Tangent) defined in the head of the trajectory data file. If the first element is a line, this parameter has no effect. If the first element is an arc, the entry angle is the tangent to the first point of the arc. Each trajectory element is identified by a number, starting from 1. The references for synchronizing external events with the trajectory execution are the starting and ending points of these elements.

Line and arc elements can be sequenced in any order. An arc is automatically placed by the controller so that its entry angle corresponds to the exit angle of the preceding element to ensure the continuity of the trajectory. But with every line segment, the user must choose the (X,Y) end-point in that way that the angle discontinuity to the previous segment does not exceed the maximum allowed angular discontinuity. The angular discontinuity is measured in degrees and is defined in the head of the trajectory data file. In theory, a trajectory can be defined only by straight lines, if two adjacent line segments have an angular difference smaller than the allowed angle of discontinuity, as shown in the Figure 26.



Figure 26: Contouring with linear lines only.

In practice this is not recommended since each angle of discontinuity corresponds to an instantaneous velocity change on both axes, which produces large accelerations. This can result in a shock to the stages and an increase in the following error. The larger the angle of discontinuity, the larger the shock and following error will be. Special consideration must be given to both these effects when increasing the maximum discontinuity angle from its default value.

8.1.5 Define Lines

A line element is defined by specifying the (X_i, Y_i) ending point.

The succeeding element's starting point is always the end point of the previous segment (X_{i-1}, Y_{i-1}).

Note that all line element positions are defined relative to the trajectory's starting point (0, 0).

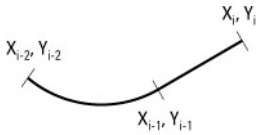


Figure 27: Line element to (X_i, Y_i) position coordinates.

As described before, when adding a new line element, the user must make sure that the discontinuity angle between the new segment and the previous one is not excessive.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8.1.6 Define Arcs

An arc is defined by specifying the radius R and the sweep angle A (Figure 28).

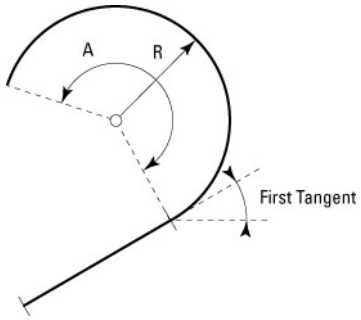


Figure 28: An arc defined with radius and angle.

Both radius and sweep angles are expressed in double precision floating point numbers. The sweep angle can range from 10^{-14} to 1.797 x 10^{308} allowing a definition of arcs from a fraction of a degree to practically an infinite number of overlapping circles.

8.1.7 Trajectory File Description

The Line-arc trajectory is defined in a file that has to be stored in the

..\public\trajectories folder of the XPS controller. This file must have the following structure:

The first line sets the “FirstTangent”: Defines the tangent angle for the first point in case of an arc. This parameter has no effect if the first element is a line.

The second line sets the “DiscontinuityAngle”: Defines the maximum allowed angle of discontinuity.

The third line must be empty for better readability.

The following lines define the Line-arc trajectory: Each line defines an element of the trajectory.

An element can be a “Line” or an “Arc”:

Line: Define X and Y positions to build a linear segment Line = X, Y.

Arc: Define radius and sweep angle to build an arc of circle Arc = R, A.

8.1.8 Trajectory File Examples

The following is an example of a trajectory file that represents a rectangle with rounded corners and with the end point equal to the starting point:

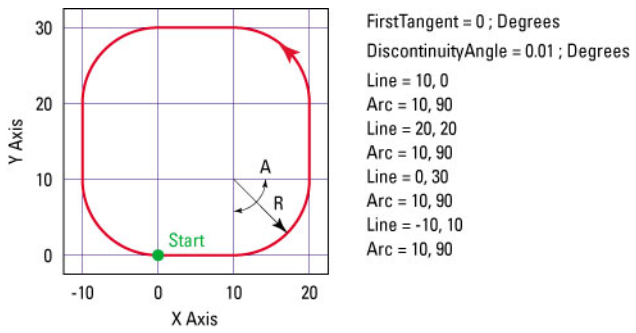


Figure 29: Graphical display of the first Line-arc trajectory data file example.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following is an example of a trajectory file that represents a rectangle with rounded corners and with the end point equal to the starting point:

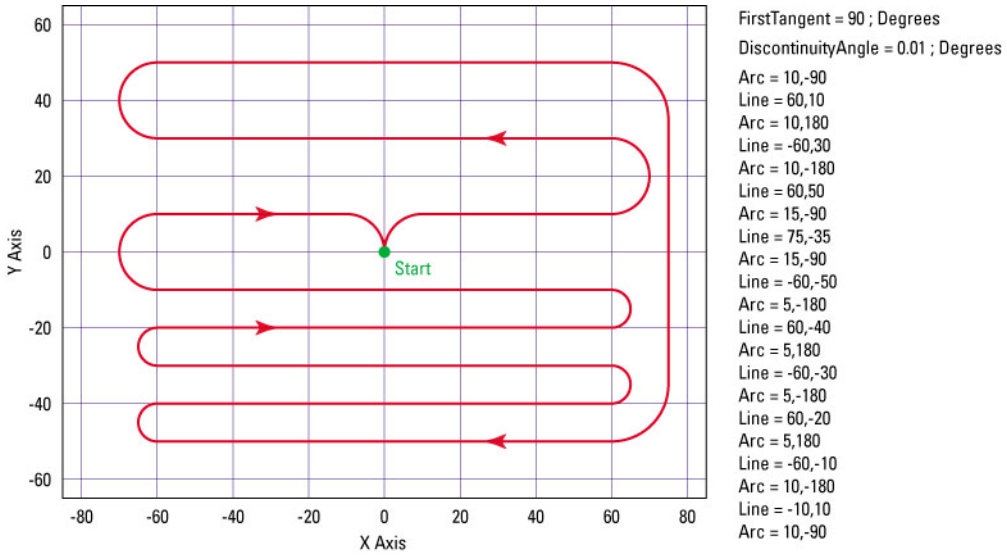


Figure 30: Graphical display of the second Line-arc trajectory data file example.

8.1.9 Trajectory Verification and Execution

There are four functions to verify or execute a Line-arc trajectory:

- **XYLineArcVerification():** Verifies a Line-arc trajectory data file.
- **XYLineArcVerificationResultGet():** Returns the last trajectory verification results, actuator by actuator. This function works only after an XYLineArcVerification().
- **XYLineArcExecution():** Executes a trajectory.
- **XYLineArcParametersGet():** Returns the trajectory's current execution parameters. This function works only while executing the trajectory.

The **function XYLineArcVerification()** can be executed at any time and is independent from trajectory execution. This function performs the following:

- Checks the trajectory file for data and syntax coherence.
- Calculates the trajectory limits, which are: the required travel per positioner, the maximum possible trajectory velocity and the maximum possible trajectory acceleration. This function defines the parameters for trajectory execution.
- If all is OK, it returns an "OK" (0). Otherwise, it returns a corresponding error.

The function **XYLineArcVerificationResultGet()** can be executed only after an XYLineArcVerification() and returns the following:

- Travel requirement in positive and negative direction for each positioner.
- The maximum possible trajectory velocity (speed) that is compatible with all positioner's velocity parameters. It returns a value for the trajectory velocity, that when applied, at least one of the positioners will reach its maximum allowed speed at least once along the trajectory. So the returned value varies between Min $\{V_{max_actuator}\}$ and $velocity = \sqrt{2 \cdot PositionerMaximumVelocity^2}$. However, this value does not take into account the positioners's acceleration, which can also limit the trajectory velocity. For example, the case of a Line-arc trajectory containing arc segments with a small radius.
- The maximum possible trajectory acceleration that is compatible with all positioners' parameters. This means that one of the positioners will reach its maximum allowed acceleration during the trajectory execution.

The XYLineArcVerificationResultGet() function returns the trajectory execution limits that have previously been calculated by the XYLineArcVerification function. Note about this function's result: Only the returned travel requirements are specific for each

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

positioner. The returned velocity/acceleration values are the same for all positioners, because they represent the trajectory's velocity/acceleration.

To execute a Line-arc trajectory, send the function XYLineArcExecution() with the parameters for the trajectory velocity, and the trajectory acceleration that is used during the start and end of the trajectory. The motion profile for Line-arc trajectories is trapezoidal. The function XYLineArcExecution() does not verify the trajectory coherence or geometric conditions (exceeding any positioners, min. or max. travel, speed or acceleration) before execution, so users must pay attention when executing a trajectory and verify the trajectory relative to the maximum possible values or possible interference. In case of an error during execution, because of bad data or because of a following error (for example if the trajectory acceleration or speed was set too high) the motion group will make an emergency stop and will enter the disabled state. The parameters for trajectory velocity and trajectory acceleration can also be set to zero. In this case the controller uses executable default values which are Min{All $V_{max_actuator}$} for trajectory velocity and Min{All $A_{max_actuator}$} for trajectory acceleration.

A trajectory can be executed many times (up to 2^{31} times) by specifying the ExecutionNumber parameter with the XYLineArcExecution function. In this case, the second run of the trajectory is simply appended to the end of the first run, while the end position of the first run is taken as a new start position (referenced to zero) of the second run. The trajectory endpoint does not need to be the same as the start point. The total trajectory is executed without stopping between the different runs.

Finally, the function **XYLineArcParametersGet()** returns the trajectory execution status with trajectory name, trajectory velocity, trajectory acceleration and current executed trajectory element. This function returns an error if the trajectory is not executing.

8.1.10 Examples of the Use of the Functions

XYLineArcVerification (XYGroup, Linearc1.trj)

This function returns a 0 if the trajectory is executable.

XYLineArcVerificationResultGet (XYGroup.XPositioner, *Name, *NegTravel, *PosTravel, *MaxSpeed, *MaxAcceleration)

This function returns the name of the trajectory checked with the last sent function XYLineArcVerification to that motion group (Linearc1.trj), the negative or left travel required for the XYGroup.XPositioner, the positive or right travel required for the XYGroup.XPositioner, the maximum trajectory velocity and the maximum trajectory acceleration.

XYLineArcExecution (XYGroup, Linearc1.trj, 10, 100, 2)

Executes the trajectory Linearc1.trj with a trajectory velocity of 10 units/s and a trajectory acceleration of 100 units/s^2 two (2) times.

XYLineArcParametersGet (XYGroup, *FileName, *TrajectoryVelocity, *TrajectoryAcceleration, *ElementNumber)

Returns the name of the trajectory in execution (Linearc1.trj), the trajectory velocity (10), the trajectory acceleration (100) and the number of the current executed trajectory element.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8.2 Splines

8.2.1 Trajectory Terminology

Trajectory: Continuous multidimensional motion path. Spline trajectories are defined in a three-dimensional XYZ space. They are available with XYZ groups only. The major benefit provided by a spline trajectory is to hit all points (except for the first and the last point that are needed to define the start and the end) and to maintain an almost constant speed (speed being the scalar of the vector velocity) throughout the entire path (except during the acceleration and deceleration periods). Please note that the trajectory speed can vary in some areas depending on the distribution of the reference points. This is related to the spline algorithm used.

Trajectory element (segment): An element of a spline trajectory is defined by a 3rd order polynomial curve joining two consecutive control points.

Trajectory velocity: The tangential linear velocity (speed) along the trajectory during its execution.

Trajectory acceleration: The tangential linear acceleration used to start and end a trajectory. Trajectory acceleration and trajectory deceleration are always equal and by default.

8.2.2 Trajectory Conventions

When defining and executing a spline trajectory, a number of rules must be followed:

- The motion group must be an XYZ group.
- All trajectories must be stored in the controller's memory under ..\public\trajectories (one file for each trajectory). Once a trajectory is started, it executes in the background allowing other groups or positioners to work independently and simultaneously.
- Each trajectory must have a defined beginning and end. Endless (infinite) trajectories are not allowed. Although, N-times (N defined by user) non-stop execution of a trajectory is allowed. As the trajectory is stored in a file, the trajectory's maximum size (maximum elements number) is unlimited for practical purposes.
- Spline trajectory elements (segments) are 3rd order polynomial curve segments $S_i(u)$, joining the positions P_{i-1} (X_{i-1}, Y_{i-1}, Z_{i-1}) and P_i (X_i, Y_i, Z_i). Here "u" is the normalized time parameter that varies from 0 (corresponding to P_{i-1}) to 1 (corresponding to P_i).
- Spline trajectories form a continuous path (each segment's output position is equal to the next segment's input position), and the segment tangential angles at the connection point of any two consecutive segments are continuous, including its derivative. For reference, this discontinuity is categorized as R^1, wherein position and velocity are continuous, but not acceleration.

8.2.3 Geometric Conventions

The Spline trajectory's coordinate system is an XYZ orthogonal system.

The X-axis of this system correlates to the XPositioner, the Y-axis to the YPositioner, and the Z-axis to the ZPositioner of the XYZ group as defined in the stages.ini.

The origin of the XYZ coordinate system is in the lower left corner, with positive values up (Z), to the right (X) and forward (Y).

All angles are measured in degrees, presented as floating point numbers. Angle origin and sign follow the trigonometric convention: positive angles are measured counter-clockwise.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8.2.4 Catmull-Rom Interpolating Splines

To trace a smooth curve that links different predefined trajectory points, the intermediate points must be calculated following a mathematical model. For the sake of simplicity, in most cases this is done by a polynomial curve (polynomial interpolation). For motion systems, the resulting curve should hit all predefined points. This is called precise interpolation in contrast to approximate interpolation (like Bezier splines), where the predefined points act only as control points. Within this class of precise interpolation are:

- Global polynomial interpolation: One polynomial represents the whole trajectory. Examples are Lagrange polynomials or Newton polynomials.
- Local polynomial interpolation: Each segment that links two consecutive trajectory points has its own polynomial. The resulting curve is obtained by segment polynomial concatenation. To limit oscillations inside segments, the polynomial order is generally limited to 3 or less. This is called spline interpolation. If the polynomial order is equal to 3, it is called cubic spline interpolation.

The interpolation methods are also classified by the continuity criterion C^k. An interpolating curve has the continuity C^k if it and its derivatives up to k-degrees are continuous in all its points. The interpolating spline curves generally have C^1 or C^2 continuity.

Catmull-Rom splines are a family of **local cubic interpolating splines** where the tangent at each point p_i is calculated based on the previous p_{i-1} and the next point p_{i+1} on the spline. In case of the spline curve tension $\tau = 1/2$ (normal case), the **Catmull-Rom** spline is described by the following equation:

$$S(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \cdot \frac{1}{2} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{bmatrix}$$

Here, p_i are the coordinates of the predefined trajectory point in x, y and z (p_{xi}, p_{yi}, p_{zi}). "u" is the normalized interpolating parameter, varying from 0 (starting at p_i) to 1 (ending at p_{i+1}).

Catmull-Rom splines have a C^1 continuity (continuity up to the first derivative), local control and interpolation. **Catmull-Rom** splines have the advantage of simple calculation without matrix inversion for on-line calculations, which is a great advantage for splines with a large number of trajectory points. For this reason, the XPS controller uses the **Catmull-Rom** spline interpolation.

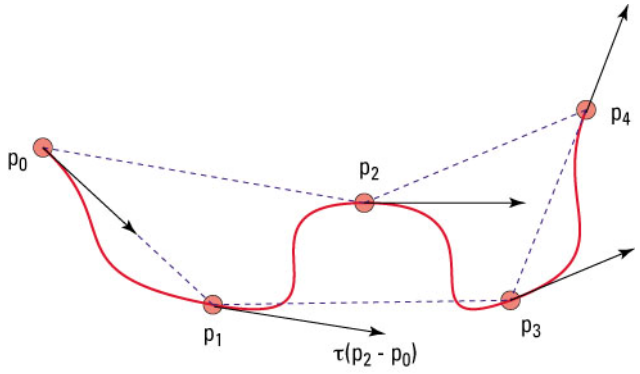


Figure 31: A Catmull-Rom spline.

8.2.5 Trajectory Elements Arc Length Calculation

Spline contouring at constant speed requires an accurate calculation of the segment's arc length. The segment's arc length can be expressed as follows:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

$$L(u_0,u_1) = \int_{u_0}^{u_1} \sqrt{\left(\frac{d}{du}Sx(u)\right)^2 + \left(\frac{d}{du}Sy(u)\right)^2 + \left(\frac{d}{du}Sz(u)\right)^2}\, du$$

Here, $u_0 = 0$ is the segment starting point and $u_1 = 1$ is the segment ending point. Sx, Sy, Sz are x-, y-, and z-components of the segment function.

This integral can only be numerically calculated, which is done by the XPS controller using the Romberg numerical integration algorithm. This guarantees that the arc length is calculated with an error less than 10^{-7} units.

8.2.6 Trajectory File Description

The spline trajectory is described in a file in the \Admin\Public\Trajectories folder of the XPS controller. Each line of this file represents one point of the spline trajectory except for the first and the last lines that are needed only to define the start and the end of the trajectory. Two consecutive points form a trajectory segment.

The format of a line in a file is:

X-Position, Y-Position, Z-Position

The separator between the X-, Y-, and Z-Position is a comma.

As mentioned before, the first and last lines of the file are needed only for the interpolation of the first and the last spline segments. These define the angle the trajectory starts and ends, but the motion system will not hit these points. So the trajectory's first "real" point (starting point) is the one defined by the second line and the trajectory's real "last" point (end point) is the one defined by the second to the last line.

The position values in the data file are relative to the physical position of the motion group at the start of the trajectory. If the position in the second line of the file (starting point) is not equal to zero (0, 0, 0), the real trajectory positions (those that the motion group will hit) are shifted further by this value.

Example

The spline trajectory file has the following format:

$$\begin{matrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ \dots & \dots & \dots \end{matrix}$$

At the moment the trajectory is executed, the motion group is at the position X_C, Y_C, Z_C. So the real matrix in absolute coordinates of the motion group is:

$$\begin{matrix} x_{c+x_0-x_1} & y_{c+y_0-y_1} & z_{c+z_0-z_1} \\ x_c & y_c & z_c \\ x_{c+x_2-x_1} & y_{c+y_2-y_1} & z_{c+z_2-z_1} \\ x_{c+x_3-x_1} & y_{c+y_3-y_1} & z_{c+z_3-z_1} \\ x_{c+x_4-x_1} & y_{c+y_4-y_1} & z_{c+z_4-z_1} \\ \dots & \dots & \dots \end{matrix}$$

8.2.7 Trajectory File Example

This trajectory example represents a spiral starting from (0, 20, 0) and ending at (0, -20, 24). As described before, the trajectory's first (-5, 19.365, -1) and last (5, -19.365, 25)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

points are only needed to define the start and end conditions of the trajectory. Because the second line (0, 20, 0) is not equal to zero (0, 0, 0), all points that the motion group will hit during the execution of the trajectory are reduced by this value from the physical starting position of the motion group.

The original data file is (except for the tabs that are only added for better readability):

-5,	19.365,	-1	-15,	13.229,	13
0,	20,	0	-10,	17.321,	14
5,	19.365,	1	-5,	19.365,	15
10,	17.321,	2	0,	20,	16
15,	13.229,	3	5,	19.365,	17
20,	0,	4	10,	17.321,	18
15,	-13.229,	5	15,	13.229,	19
10,	-17.321,	6	20,	0,	20
5,	-19.365,	7	15,	-13.229,	21
0,	-20,	8	10,	-17.321,	22
-5,	-19.365,	9	5,	-19.365,	23
-10,	-17.321,	10	0,	-20,	24
-15,	-13.229,	11	5,	-19.365,	25
-20,	0,	12			

With this data file, the real trajectory points relative to the physical start position of the motion group are (first and last lines are eliminated because the motion group will not hit these points and the values from the second column are reduced by 20 as the first line was (0, 20, 0)):

0,	0,	0	-15,	-6.771,	13
5,	-0.635,	1	-10,	-2.679,	14
10,	-2.679,	2	-5,	-0.635,	15
15,	-6.771,	3	0,	0,	16
20,	-20,	4	5,	-0.635,	17
15,	-33.229,	5	10,	-2.679,	18
10,	-37.321,	6	15,	-6.771,	19
5,	-39.365,	7	20,	-20,	20
0,	-40,	8	15,	-33.229,	21
-5,	-39.365,	9	10,	-37.321,	22
-10,	-37.321,	10	5,	-39.365,	23
-15,	-33.229,	11	0,	-40,	24
-20,	-20,	12			

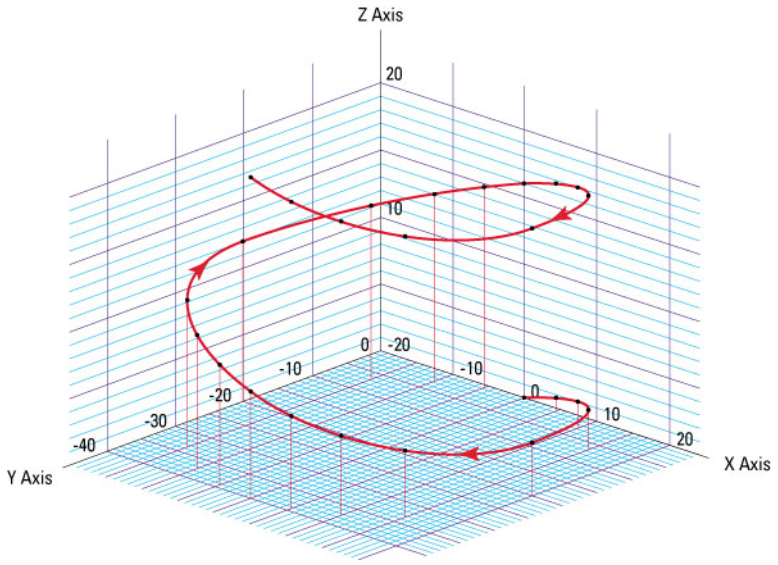


Figure 32: Executing the above normalized trajectory data file with the Catmull-Rom spline algorithm.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8.2.8 Spline Trajectory Verification and Execution

Here are four functions to verify or execute a spline trajectory:

- **XYZSplineVerification():** Verifies a spline trajectory data file.
- **XYZSplineVerificationResultGet():** Returns the last trajectory verification results, actuator by actuator. This function works only after an XYZSplineVerification().
- **XYZSplineExecution():** Executes a trajectory.
- **XYZSplineParametersGet():** Returns the trajectory current execution parameters. This function works only while executing of the trajectory.

The function **XYZSplineVerification()** can be executed at any moment and is independent from the trajectory execution. This function performs the following:

- Checks the trajectory file for data and syntax coherence.
- Calculates the trajectory limits, which are the required travel per positioner, the maximum possible trajectory velocity and the maximum possible trajectory acceleration. This function defines the parameters for trajectory execution.
- If all is OK, it returns an "OK" (0). Otherwise, it returns a corresponding error.

The function **XYZSplineVerificationResultGet()** can be executed only after an XYZSplineVerification() and returns the following:

- Travel requirement in the positive and negative directions for each positioner.
- The maximum possible trajectory velocity (speed) that is compatible with all positioners' velocity parameters. It returns a value for the trajectory velocity, that when applied, at least one of the positioners will reach its maximum allowed speed at least once along the trajectory. So, the returned value varies between Min{Vmax_actuator} and $velocity = \sqrt{\sum PositionerMaximumVelocity^2}$. However, this value does not take into account that the positioners' acceleration can limit the trajectory velocity. This is the case with splines that contain sharp curved segments.
- The maximum trajectory acceleration that is compatible with all positioner parameters. At this trajectory acceleration, one of the positioners will reach its maximum allowed acceleration during trajectory execution.

The function **XYZSplineVerificationResultGet()** returns the trajectory execution limits that have previously been calculated by the XYZSplineVerification function. Note on this function's response: Only the returned travel requirements are specific for each positioner, the returned velocity/acceleration values are the same for all positioners, because they represent the trajectory's velocity/acceleration.

To execute a spline trajectory, send the function **XYZSplineExecution()** with the parameters for the trajectory velocity and the trajectory acceleration (the trajectory acceleration that is used during the start and the end of the trajectory). The motion profile for spline trajectories is trapezoidal. The function XYZSplineExecution() does not verify the trajectory's coherence or geometric conditions (exceeding any positioner's min. or max. travel, speed or acceleration) before execution, so users must pay attention when executing a trajectory without verifying the trajectory the maximum possible values. In case of an error during execution, because of bad data or because of a following error (for example the trajectory acceleration or speed was set too high) the motion group will make an emergency stop and will go to the disabled state. The parameters for trajectory velocity and trajectory acceleration can also be set to zero. In this case the controller uses executable default values which are the Min{All $V_{max_actuator}$} for trajectory velocity and Min{All $A_{max_actuator}$} for trajectory acceleration.

Finally, the function **XYZSplineParametersGet()** returns the trajectory execution status with trajectory name, trajectory velocity, trajectory acceleration and current executed trajectory element. This function returns an error if the trajectory is not executing.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8.2.9 Examples

XYZSplineVerification (XYZGroup, Spline1.trj)

This function returns a 0 if the trajectory is executable.

XYZSplineVerificationResultGet (XYZGroup.XPositioner, *Name, *NegTravel, *PosTravel, *MaxSpeed, *MaxAcceleration)

This function returns the name of the trajectory checked with the last sent function XYZSplineVerification to that motion group (Spline1.trj), the negative travel required for the XYZGroup.XPositioner, the positive travel required for the XYZGroup.XPositioner, the maximum trajectory velocity and the maximum trajectory acceleration.

XYZSplineExecution (XYZGroup, Spline1.trj, 10, 100)

Executes the trajectory Spline1.trj with a trajectory velocity of 10 units/s and a trajectory acceleration of 100 units/s^2.

XYZSplineParametersGet (XYZGroup, *FileName, *TrajectoryVelocity, *TrajectoryAcceleration, *ElementNumber)

Returns the name of the trajectory being executed (Spline1.trj), the trajectory velocity (10), the trajectory acceleration (100) and the number of the currently executed trajectory element.

8.3 PVT Trajectories

8.3.1 Trajectory Terminology

Trajectory: continuous multidimensional motion path. PVT stands for Position, Velocity, and Time. PVT trajectories are defined in an n-dimensional space (n = 1 to 8). These are available with MultipleAxes groups. A PVT trajectory is generated with continuous movements of the MultipleAxes group's positioners over several time periods. For each period, each positioner must complete a defined displacement from its current position and a defined output velocity at the end of the period. By definition, there is no constant vector velocity and no definition for a vector acceleration in contrast to Line-arc trajectories or splines.

Trajectory element (segment): An element of a PVT trajectory is defined by a set of all positioner displacements and output velocities and the duration for the segment. In the PVT data file, each element is represented by a line of values:

DT, DP1, VO1, DP2, VO2, ... DPn, VOn

DT:	The segment duration in seconds.
DP1, DP2,…, DPn:	Positioners' (#1, #2,…, #n) displacements during DT.
VO1, VO2,…, VOn:	Positioners' output velocities at the end of DT.

8.3.2 Trajectory Conventions

When defining or executing a PVT trajectory, a number of rules must be followed:

- The motion group must be a MultipleAxes group.
- All trajectories must be stored in the controller's memory in ..\Public\Trajectories. Once a trajectory is started, it executes in the background allowing other groups to work independently and simultaneously.
- Each trajectory must have a beginning and an end. Endless (infinite) trajectories are not allowed. Although, N-times (N defined by user) non-stop execution of a trajectory is allowed. Since the trajectory is stored in a file, the trajectory's maximum size (maximum elements number) is practically not limited.
- PVT trajectory elements (segments) are 3rd order polynomial pieces for each positioner that hit the positions P_{i-1} (at time t_{i-1} with a velocity v_{i-1}) and positions P_i

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

(at time t_i with a velocity v_i). There is no direct link between the trajectories of the different positioners in a MultipleAxes group.

- PVT trajectories form a continuous path (each segment output position is equal to the next segment input position), and the segment tangential angles at the connection point of any two consecutive segments are continuous including its derivative. It means that the PVT trajectory continuity property is R^1.
- The input velocity of any element is equal to the output velocity of the previous element. The input velocity for the first element is always zero. The output velocity of the last element must be zero as well.

8.3.3 Geometric Conventions

- The coordinate system can be any convention, it does not need to be an orthogonal system.
- A PVT trajectory can be defined for any MultipleAxes group. There is no limit to the number of positioners belonging to that MultipleAxes group. It is also possible to define a PVT trajectory for a MultipleAxes group that contains only one positioner.

8.3.4 PVT Interpolation

For each positioner belonging to the MultipleAxes group, the PVT trajectory calculates a 3rd order polynomial curve P(u) that can be presented by the following equations:

Profile coefficient

- Acceleration jerk:

$$\text{Jerk} = \frac{6 \cdot \left[\text{DT} \cdot (V_{in} + V_{out}) - 2 \cdot \text{DX}\right]}{\text{DT}^3}$$

- Initial acceleration:

$$G_{in} = \frac{2 \cdot \left[3 \cdot \text{DX} - \text{DT} \cdot (2 \cdot V_{in} + V_{out})\right]}{\text{DT}^2}$$

- Final acceleration:

$$G_{out} = \frac{2 \cdot \left[\text{DT} \cdot (V_{in} + 2 \cdot V_{out}) - 3 \cdot \text{DX}\right]}{\text{DT}^2}$$

Profile equation

- Acceleration:

$$\text{Acc}(t) = G_{in} + \text{Jerk} \cdot t$$

- Velocity:

$$\text{Vel}(t) = V_{in} + G_{in} \cdot t + \frac{\text{Jerk} \cdot t^2}{2}$$

- Position:

$$\text{Pos}(t) = V_{in} \cdot t + \frac{G_{in} \cdot t^2}{2} + \frac{\text{Jerk} \cdot t^3}{6}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Here:

DT is the segment duration in seconds

DX is the displacement during DT

V_{in} is the output velocity of the previous segment (which is equal to the input velocity of the current segment)

V_{out} is the output velocity of the current segment.

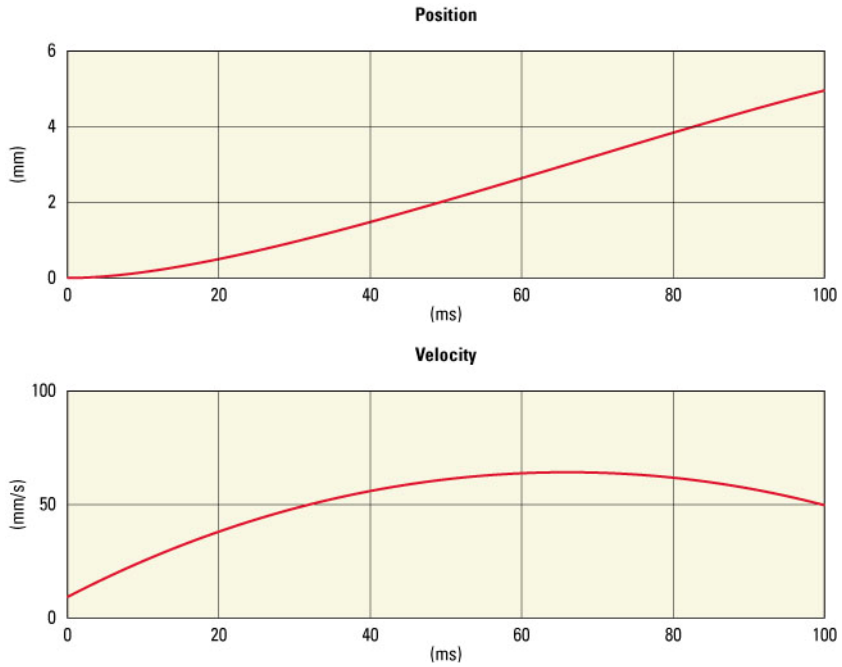
t is the time in seconds starting at 0 (entry of the current element) and ending at DT (end of the segment)

8.3.5 Influence of the Element Output Velocity to the Trajectory

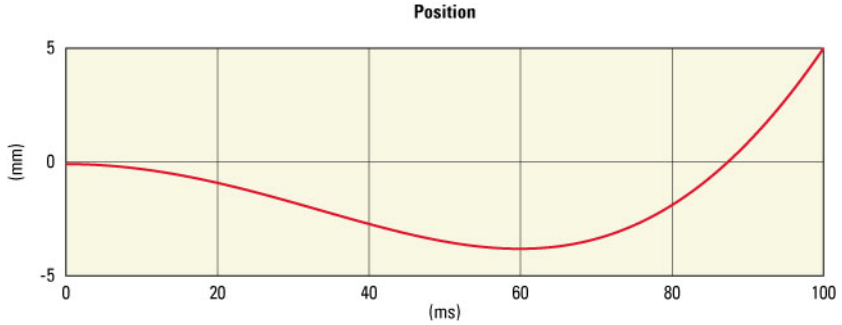
The contour of each PVT trajectory element is influenced not only by the displacement, but also by the input and output velocities. As the user decides on these velocities, attention must be placed on these values to get the desired results.

The effect of the velocity is illustrated in the following example which shows the position and velocity profiles for one segment of a PVT trajectory that has a displacement of 5 mm, a duration of 100 ms, an input velocity of 10 mm/s and an output velocity of either 50 mm/s or 500 mm/s:

- If the output velocity is equal to 50 mm/s.



- If the output velocity is equal to 500 mm/s.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

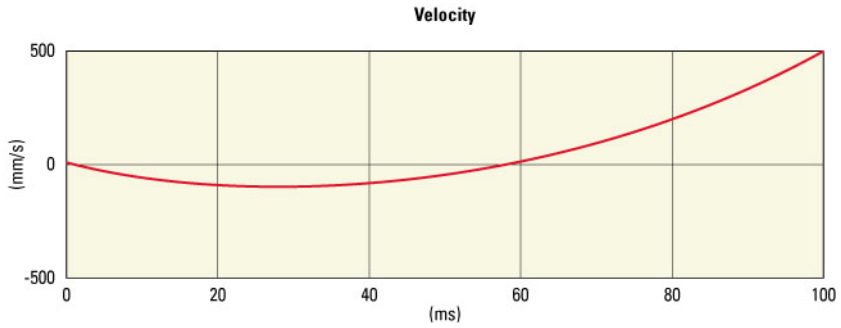


Figure 33: PVT trajectory element in execution: the comparison.

A PVT trajectory must have three parameters: position, velocity and time. With a given target displacement, output velocity and time duration, the PVT trajectory calculates intermediate positions and velocities as a function of time.

With an output velocity of 50 mm/s, the positioner has "enough" time to achieve the displacement within the assigned time (100 ms) in the forward direction. The velocity increases at the beginning and then slows down towards the end. The position always increases up to the target position (5 mm).

On the other hand, when the output velocity is set to 500 mm/s, the positioner does not have enough time to achieve the displacement and speed output required in the forward direction. So the positioner will first reverse the direction of motion to be able to approach the end position with a speed of 500 mm/s.

8.3.6 Trajectory File Description

The PVT trajectory is described in a file that is in the ..\public\trajectories folder of the XPS controller. Each line of this file represents one element of the trajectory.

A line contains several data separated by a comma. The number of data in each line depends on the number of positioners in the MultipleAxes group. The first data in each line is the duration of the element. The following data is grouped in pairs of two representing the displacement and the output velocity for each positioner of the group.

So the line format is as follows:

Data #1: Element duration (seconds).

Data #2: 1st positioner's displacement (units).

Data #3: 1st positioner's output velocity (units/s).

Data #4: 2nd positioner's displacement (units).

Data #5: 2nd positioner's output velocity (units/s).

(And so on…)

NOTE

The first positioner is always the first defined in the system.ini of the MultipleAxes group (see ActuatorInUse), the second positioner is always defined as second, and so on…

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8.3.7 Trajectory File Example

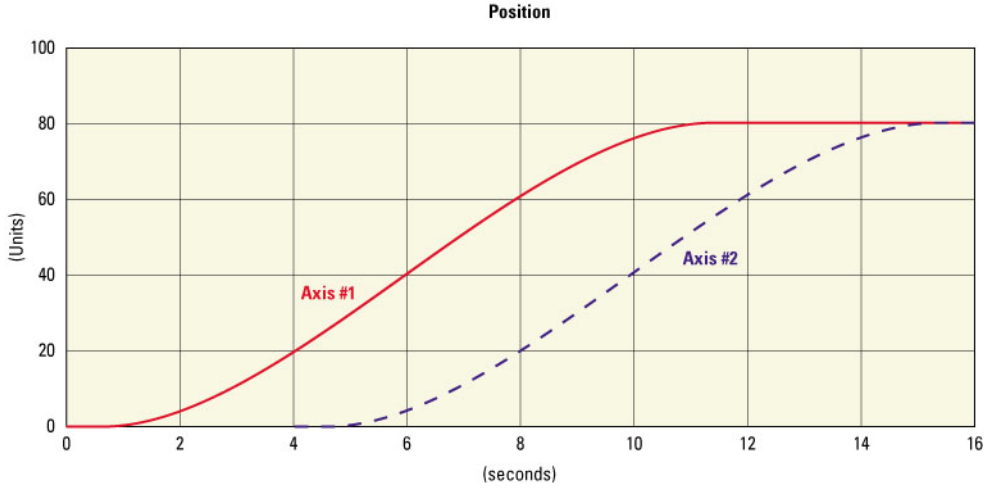
Following is an example of a PVT trajectory defined in a MultipleAxes group that contains two positioners. The tabs are added for better readability and are ignored in a line:

```
1.0,    0.4167,    1.25,    0,        0
1.0,    2.9167,    5,       0,        0
1.0,    7.0833,    8.75,    0,        0
1.0,    9.5833,    10,      0,        0
1.0,    10,        10,      0.4167,   1.25
1.0,    10,        10,      2.9167,   5
1.0,    10,        10,      7.0833,   8.75
1.0,    10,        10,      9.5833,   10
1.0,    9.5833,    8.75,    10,       10
1.0,    7.0833,    5,       10,       10
1.0,    2.91667,   1.25,    10,       10
1.0,    0.41667,   0,       10,       10
1.0,    0,         0,       9.5833,   8.75
1.0,    0,         0,       7.0833,   5
1.0,    0,         0,       2.91667,  1.25
1.0,    0,         0,       0.41667,  0
```

This file represents the following data:

Time Period (s)	Axis #1 Displacement	Axis #1 Velocity Out	Axis #2 Displacement	Axis #2 Velocity Out
1.0	0.4167	1.25	0	0
1.0	2.9167	5.0	0	0
1.0	7.0833	8.75	0	0
1.0	9.5833	10	0	0
1.0	10	10	0.4167	1.25
1.0	10	10	2.9167	5
1.0	10	10	7.0833	8.75
1.0	10	10	9.5833	10
1.0	9.5833	8.75	10	10
1.0	7.0833	5	10	10
1.0	2.9167	1.25	10	10
1.0	0.4167	0	10	10
1.0	0	0	9.5833	8.75
1.0	0	0	7.0833	5
1.0	0	0	2.9167	1.25
1.0	0	0	0.4167	0

Table 1: The trajectory data file.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

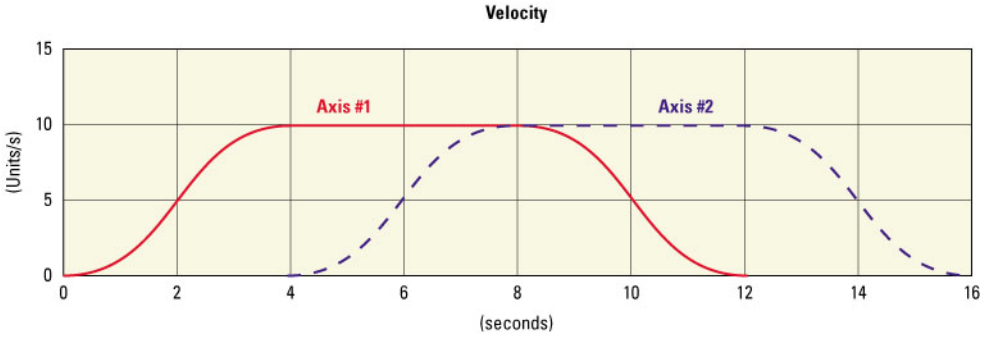


Figure 34: Executing the trajectory data file with the PVT algorithm.

8.3.8 PVT Trajectory Verification and Execution

Here are four functions to verify or execute a PVT trajectory:

- **MultipleAxesPVTVerification():** Verifies a PVT trajectory data file.
- **MultipleAxesPVTVerificationResultGet():** Returns the results of the last trajectory verification call, actuator by actuator. This function works only after a MultipleAxesPVTVerification().
- **MultipleAxesPVTExecution():** Executes a PVT trajectory.
- **MultipleAxesPVTParametersGet():** Returns the trajectory's current execution parameters. This function works only while executing a trajectory.

The function **MultipleAxesPVTVerification()** can be executed at any moment and is independent of the trajectory execution. This function does the following:

- Checks the trajectory file for data and syntax coherence.
- Simulates the trajectory to determine the positioner's travel requirements in negative and positive directions and the maximum allowed speed and acceleration for each positioner. This function determines whether the trajectory is executable.
- If all is OK, it returns an "OK" (0). Otherwise it returns a corresponding error. An error for instance is reported if one of the positioner's speed or acceleration reached during the trajectory exceeds the maximum allowed speed or acceleration.

The function **MultipleAxesPVTVerificationResultGet()** can be executed only after a MultipleAxesPVTVerification(). It returns the trajectory limits for each positioner, which are the travel requirements in positive and negative directions, the achieved maximum speed and acceleration.

To execute a PVT trajectory, send the function **MultipleAxesPVTExecution()** while specifying the file name and the number of cycles. This function does not verify the trajectory's coherence or geometric conditions (exceeding any positioner's min. or max. travel, speed or acceleration) before execution, so users must be careful when executing a trajectory without verifying the trajectory first. In case of an error during execution, because of bad data or because of a following error, the motion group will make an emergency stop and will go to the disabled state.

Finally, the function MultipleAxesPVTParametersGet() returns the trajectory name and the number of the trajectory element that is currently being executed. This function returns an error if the trajectory is not executing.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8.3.9 Examples of the Use of the functions

MultipleAxesPVTVerification (NGroup, PVT1.trj)

This function returns a 0 if the trajectory is executable.

MultipleAxesPVTVerificationResultGet (NGroup.1Positioner, *Name, *NegTravel, *PosTravel, *MaxSpeed, *MaxAcceleration)

This function returns the name of the trajectory verified with the last functions call of MultipleAxesPVTVerification to the motion group NGroup (PVT1.trj) and the trajectory limits for the positioner NGroup.1Positioner. These trajectory limits are: the negative or left travel requirement, the positive or right travel requirement, the achieved maximum speed and acceleration. Make sure that these trajectory limits (required negative and positive travel, speed and acceleration) are within the soft limits of the stages defined in the stages.ini file (section Travel: MinimumTargetPosition, MaximumTargetPosition and section Profiler: MaximumVelocity, MaximumAcceleration).

MultipleAxesPVTExecution (NGroup, PVT1.trj, 5)

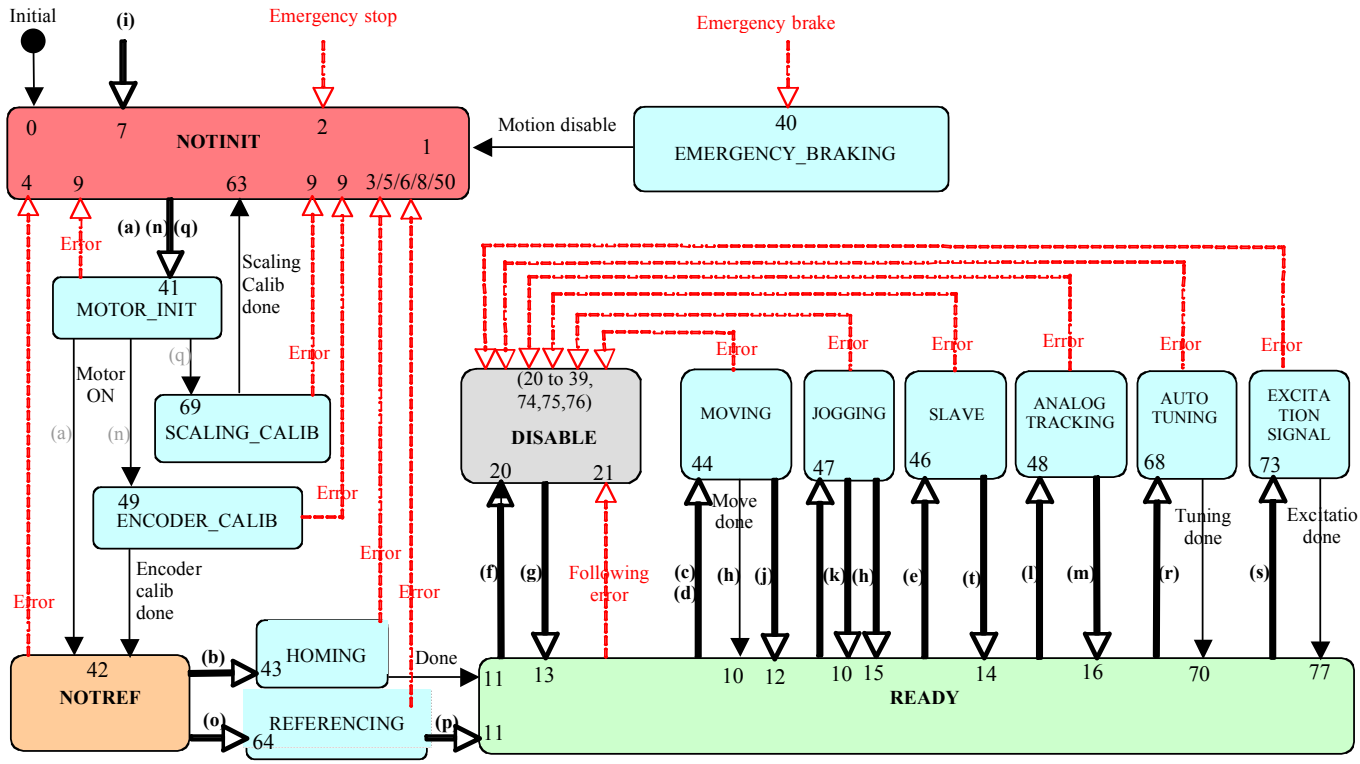
Executes the trajectory PVT1.trj five (5) times.

MultipleAxesPVTParametersGet (NGroup, *FileName, *ElementNumber)

Returns the currently executed trajectory file name (PVT1.trj) and the number of the currently executed trajectory element.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

9.0 Emergency Brake and Emergency Stop Cases



Note

Emergency brake brings a stage to a stop, then sets the motor power to Off. Emergency stop: sets motor power to Off only.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Emergency Brake occurs when:

Case	Error
Standard end of run driver safety supervisor Standard limit and home encoder safety supervisor Standard limit and limit encoder safety supervisor	• Plus end of run is detected • Minus end of run is detected
Line arc trajectory execution	• Error occurs when reading or getting trajectory parameters • The user target position is outside the MinimumTargetPosition and MaximumTargetPosition value • Actual positioner velocity is greater than the *MaximumVelocity* value
Spline trajectory execution	• Error occurs when reading or getting trajectory parameters • The user target position is outside the *MinimumTargetPosition* and *MaximalTargetPosition* value • Actual positioner velocity is greater than the *MaximumVelocity* value
PVT trajectory execution	• Error occurs when reading or getting trajectory parameters • Error occurs during trajectory execution • The user target position is outside the *MinimumTargetPosition* and *MaximalTargetPosition* values • Actual positioner velocity is greater than the *MaximumVelocity* value
S-gamma motion of a slave or a gantry secondary positioner	• Group positioner is not in the gantry home process, • **And** end of run detection is enabled • **And** the group is not a spindle group • **And** the user target position is outside the *MinimumTargetPosition* and *MaximalTargetPosition* value

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Emergency Stop occurs when:

Case	Error
AquadBEncoder fault	• Quadrature error • FOC fault (signals noisy or too fast)
Analog interpolator encoder fault	• Quadrature error • FOC fault • Hard interpolator quadrature fault • Hard interpolator fault (IP200)
N1231BEncoder fault (3 axes)	• Signal error • Glitch error
Analog interpolated theta encoder fault	• Quadrature error • FOC fault • Hard interpolator quadrature fault • Hard interpolator fault (IP200) • XY correction fault
AquadBTheta encoder	• Quadrature error • FOC fault • XY correction fault
AnalogAccelerationMotorInterface AnalogDualSinAccelerationMotorInterface AnalogPositionMotorInterface AnalogSinAccelerationMotorInterface AnalogStepperPositionMotorInterface AnalogVelocityMotorInterface AnalogVoltageMotorInterface DigitalStepperPositionMotorInterface AnalogSinAccelerationLMIMotorInterface AnalogAccelerationTZMotorInterface AnalogPositionPiezoMotorInterface	• Driver fault
Single Axis with clamping control Single Axis theta	• Unclamped state

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10.0 Compensation

The XPS controller features different compensation methods that improve the performance of a motion system:

Backlash compensation: The use of backlash compensation improves the bi-directional repeatability and accuracy of a motion device that has mechanical play. Backlash compensation is applicable to all positioners, but it is not available in all motion modes. When backlash compensation is activated, the XPS controller adds a user-defined BacklashValue to the TargetPosition to calculate a new target position whenever the direction of motion reverses. This internally used new target position is then the basis for the calculations of the motion profiler. No modification of the actual target is performed.

Linear error compensation: The linear error compensation helps improve the accuracy of a motion device by eliminating linear error sources. Linear errors can be caused by screw pitch errors, linear increasing angular deviations (abbe errors), thermal effects or cosine errors (misalignment between the feedback device and the direction of motion). Linear error compensation is applicable to all positioners. Its value is defined in the stages.ini. When set to other than zero, the encoder positions are compensated by this value. Linear error compensation can be used in conjunction with other compensation. For this reason, keep in mind the effects of using linear error compensation in addition to other compensation methods.

Positioner mapping: In contrast to the linear error compensation, positioner mapping also allows compensation for nonlinear error sources. Positioner mapping is done by sending a compensation table to the XPS controller and configuring the needed settings in the stages.ini. Positioner mapping is available with all positioners and works in parallel with other compensations except for the backlash compensation method. Better accuracy performance is achievable with linear compensation and positioner mapping combined.

XY mapping: XY mapping is only available with XY groups. It allows compensation for all errors of an XY group at any position of the XY group by sending two compensation tables to the XPS controller (x and y compensations mapped to x and y positions). The XY mapping is dynamically taken into account on the corrector loop of the XPS controller. XY mapping works in parallel to other compensation methods. Keep in mind that the results of XY mapping may not be the same as those of Positioner mapping or linear compensation alone.

XYZ mapping: XYZ mapping is only available with XYZ groups. It compensates for all errors of an XYZ group at any position of the XYZ group by sending three compensation files to the XPS controller (x compensations mapped to x, y, and z positions, and so on). The XYZ mapping is dynamically taken into account on the corrector loop of the XPS controller. XYZ mapping works in parallel to other compensation methods. Keep in mind that the results of XYZ mapping may not be the same as those of Positioner mapping or linear compensation alone.

TargetPosition, SetpointPosition & CurrentPosition are accessible via function and Gathering (Data Collection).

SetpointVelocity, SetpointAcceleration & FollowingError are accessible via Gathering (Data Collection).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

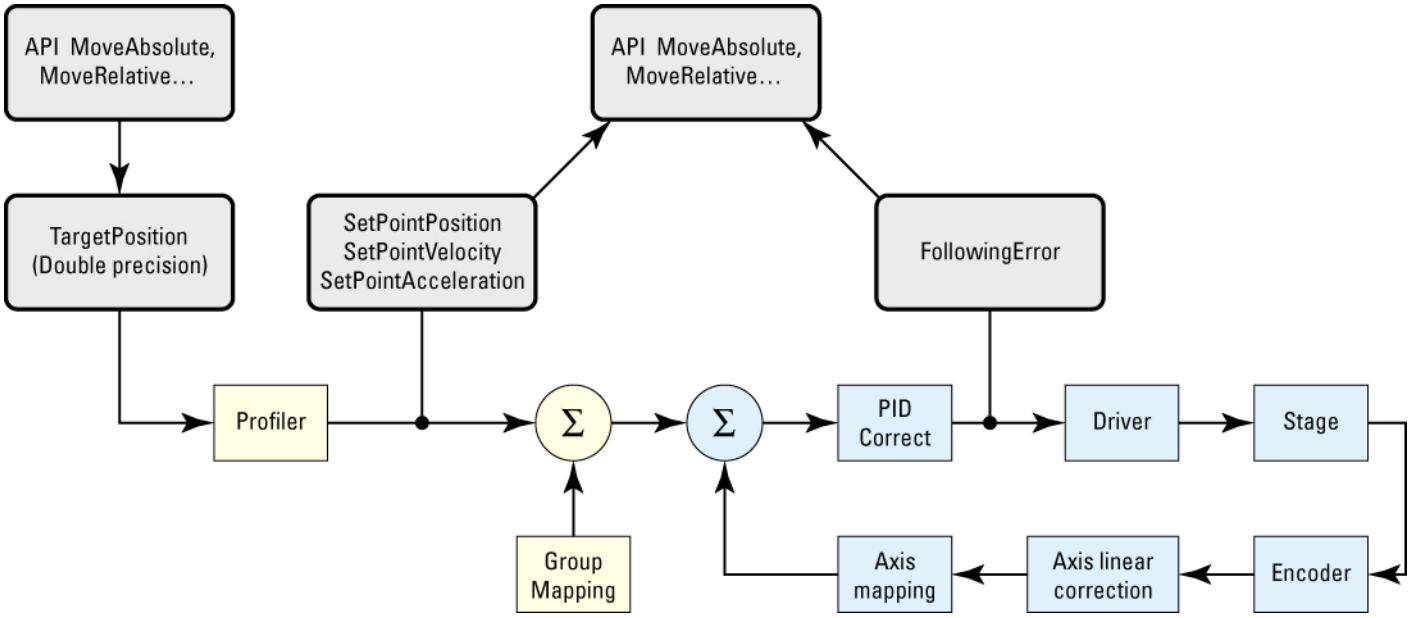


Figure 35: Definition of different positions for one actuator.

10.1 Backlash Compensation

Backlash compensation is applicable on all positioners, but works only under certain conditions:

- The "HomeSearchSequenceType" in the stages.ini must be different from "CurrentPositionAsHome".
- Backlash compensation is not compatible with positioner mapping. So for positioners with backlash compensation, it is not allowed to have an entry for "PositionerMappingFileName" in the stages.ini.
- Backlash compensation is not compatible with trajectories (Line-Arc, Spline, PVT), jog or analog tracking. So it is not possible to execute any trajectory, to use the jog mode or to enable the analog tracking with any motion group that contains positioners with backlash compensation enabled.

After the above has been taken into consideration, a number of steps need to be taken to enable backlash compensation. First of all, there must be a value larger than 0 for "backlash" in the stages.ini. But this setting does not automatically enable backlash compensation. To do so, send the function **PositionerBacklashEnable()** while the motion group, which includes the positioner is disabled. To disable backlash compensation (for instance to execute a jog motion or to use analog tracking), use the function **PositionerBacklashDisable().** The value for backlash compensation can be changed at any time with the function **PositionerBacklashSet().** The new value for the backlash will be taken into account with the next following move. Finally, the function **PositionerBacklashGet()** returns the current value of the backlash and the backlash status ("enabled" or "disabled").

For backlash setting to remain set after power down, the stages.ini file must be modified with the value desired.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

In the Backlash section of the stages.ini file, set a value greater than or equal to 0:

```
;--- Backlash
Backlash = 5                    ; units
```

This example shows the sequence of functions that enable backlash compensation:

PositionerBacklashEnable (MyGroup.MyPositioner)

GroupInitialize (MyGroup)

GroupHomeSearch (MyGroup)

…

PositionerBacklashSet (MyGroup.MyPositioner, 10)

PositionerBacklashGet (MyGroup.MyPositioner, *Backlash, *Status)

Returns the backlash value (10) and the backlash status (Enable).

…

PositionerBacklashDisable (MyGroup.MyPositioner)

10.2 Linear Error Correction

Linear error correction is applicable on all positioners and works in parallel with any other compensation. To use linear error correction, you need to set a value for "LinearErrorCorrection" in the stages.ini. When set, the corrected positions are calculated in the following way:

$$\textbf{Corrected position} = \textbf{HomePreset} + (\textbf{EncoderPosition} - \textbf{HomePreset}) \times (1 + \textbf{LinearEncoderCorrection}/10^6)$$

The value of LinearEncoderCorrection is specified in ppm (parts per million). The correction is applied relative to the physical home position of the positioner (the Encoder position by definition is set to the HomePreset value at the home position). This hardware reference for linear error correction has the advantage of being independent of the value of the HomePreset.

Example

In the Encoder section of the stages.ini file, set a value other than 0,
but $-0.5 \times 10^6 <$ value $< 0.5 \times 10^6$, in parameter LinearEncoderCorrection:

```
;--- Encoder
EncoderType =AquadB
EncoderResolution = 0.001       ; unit
LinearEncoderCorrection =5      ; ppm
```

10.3 Positioner Mapping

Positioner mapping corrects for any nonlinear errors of a positioner. Positioner mapping is applicable on all positioners and can be used with to other compensations except backlash compensation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

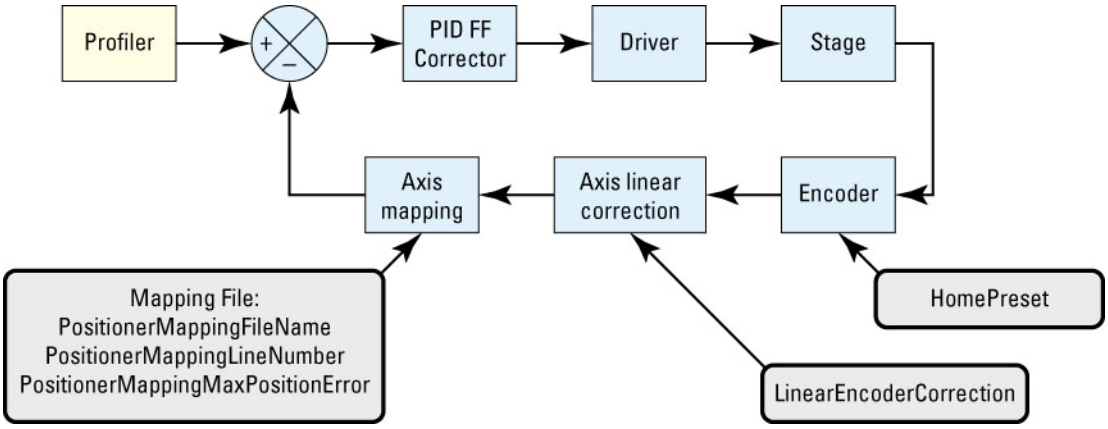


Figure 36: Positioner Mapping.

- **HomePreset:** Encoder position value at the home position.
- **LinearEncoderCorrection:** Value in ppm. Correction is given by

  CorrectedPosition = HomePreset +
  (EncoderPosition – HomePreset)*(1+LinearCorrection/10^6).

- **Mapping file:** Declaration of mapping in the stages.ini file (Positioner mapping section).

The positioner mapping data is defined in a text file. Each line of that file represents one set of data. Each set of data is composed of the position and the error at this position. The separator between the two data entries in each line is a tab. All positions are relative to the physical home position of the positioner. The data file must contain the line "0 0", which means that the error at the home position is 0. This hardware reference for positioner mapping has the advantage of being independent of the value of the HomePreset.

The following shows the general structure of such a data file:

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

To activate positioner mapping, the mapping file must be in the ..\admin\config directory of the XPS controller and the following settings must be configured in the stages.ini:

- **PositionerMappingFileName:** Name of the mapping file.
- **PositionerMappingLineNumber:** Number of lines of the file.
- **PositionerMappingMaxPositionError:** Maximum absolute error in the file must be larger than any entry in the mapping file. To be read properly, the error entries must be in index format, see example.

PositionerMappingLineNumber and PositionerMappingMaxPositionError are only used to check for the correctness of the mapping file.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

The following shows an example of a positioner mapping data file:
PosMapping.txt

-3.00	-0.00125
-2.00	-0.00112
-1.00	-0.00137
0.00	0.00000
1.00	0.00140
2.00	0.00145
3.00	0.00154

Define the positioner mapping in the stages.ini file:

```
;--- Backlash
Backlash =0                          ; unit

;--- Positioner mapping
PositionerMappingFileName = PosMapping.txt
PositionerMappingLineNumber = 7
PositionerMappingMaxPositionError = 0.00154

;--- Travels
MinimumTargetPosition =-3            ; unit
HomePreset =0                        ; unit
MaximumTargetPosition =3             ; unit
```

NOTE

These travel limits must be equal to or be within the positioner's limit positions of the mapping file (+3 and -3 in the above example).

Use of the functions:

- **GroupInitialize(MyGroup)**
- **GroupHomeSearch(MyGroup)**
- **GroupMoveAbsolute(MyGroup.Positioner, 0.25)**

The mapping file must at least cover the minimum and the maximum travel of the positioner. It must cover MinimumTargetPosition and MaximumTargetPosition parameters defined in the stages.ini, section Travels. In the example above, the travel of the positioner can not be larger than ±3 units, but it can be smaller than this. The units for the data are the same as defined by EncoderResolution in the stages.ini. The data reads as follows: the corrected position at position 3.00 units is 2.99846 units (3.00 - 0.00154). Between two data points, the XPS controller performs a linear interpolation of the error. The corrected position at position 0.25 units is 0.24965 units (0.25 - 0.00140*0.25/1).

NOTE

Mapping is a function implemented within the controller to correct positioning errors. Once activated, mapping is transparent to the user. The function GroupPositionCurrentGet doesn't return 0.24965 (0.25 - 0.00140*0.25/1) but 0.25.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10.4 XY Mapping

XY mapping is only applicable to XY groups. It compensates for all errors of an XY group at any position of that XY group. XY mapping can be used in conjunction with other compensations, including positioner mapping. So care must be taken about the unwanted effects of using XY mapping and other compensation at the same time.

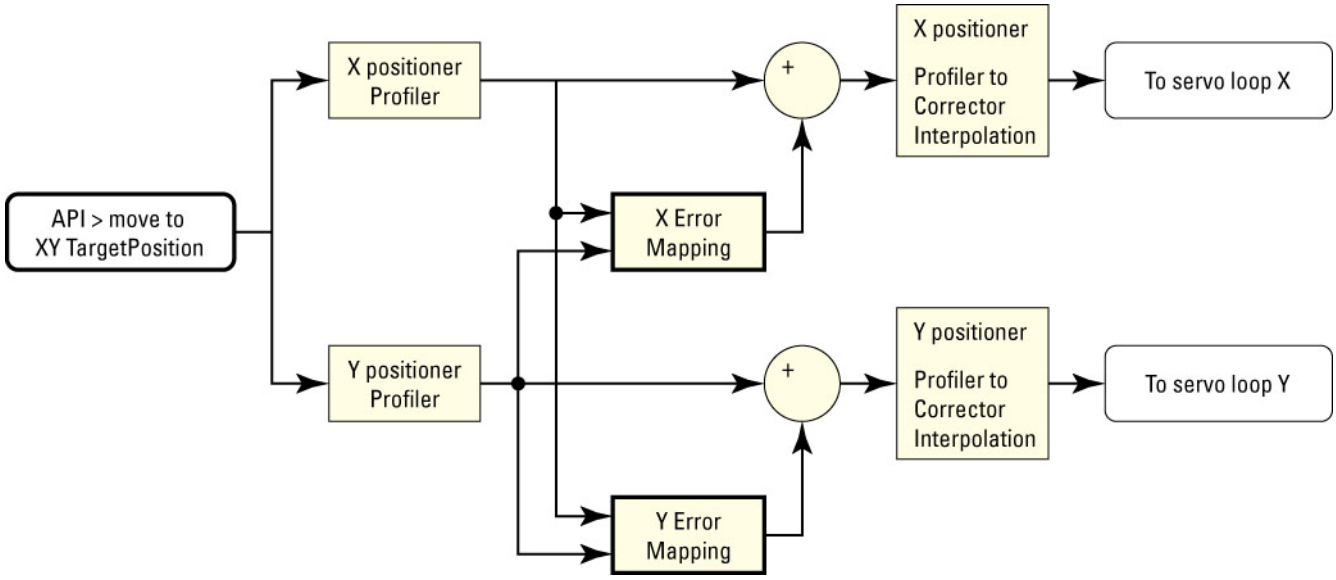


Figure 37: XY Mapping

XY mapping is defined by 2 compensation tables, in text file format, each for X and Y errors. In each of these files, the first column specifies the X positions, X being the first positioner of the XY group, and the first row, the Y positions. Each cell represents the axis error for that X,Y position as shown in the tables below. The first entry in that file must be 0 (zero). The separator between the data in each row is the tab. All positions are relative to the physical home position of the XY group. The data files must contain the X position = 0 and the Y position = 0. The error at X = Y = 0 must be 0, which means that the error at the home position is 0. This hardware reference for XY mapping has the advantage of being independent of the value of the HomePreset.

The following shows the structure of such mapping files:

X Mapping File

XMappingColumnNumber

0	Y min.	Y1	…	0	…	Y max.
X min.	X err 0 0	…	…	…	…	…
X1	…	…	…	…	…	…
…	…	…	…	…	…	…
0	…	…	…	0	…	…
…	…	…	…	…	…	…
X max.	…	…	…	…	…	…

XMappingLineNumber

Y Mapping File

YMappingColumnNumber

0	Y min.	Y1	…	0	…	Y max.
X min.	Y err 0 0	…	…	…	…	…
X1	…	…	…	…	…	…
…	…	…	…	…	…	…
0	…	…	…	0	…	…
…	…	…	…	…	…	…
X max.	…	…	…	…	…	…

YMappingLineNumber

Figure 38: XY Mapping Files.

NOTE

Error in X = Y = 0 must be 0. This value in the file corresponds to the HomePreset position in the XY group reference.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To activate XY mapping, the mapping files must be in the ..\admin\config directory of the XPS controller and the following settings must be configured in the system.ini:

- **XMappingFileName:** Name of the mapping file.
- **XMappingLineNumber:** Total number of lines of that file.
- **XMappingColumnNumber:** Total number of columns of that file.
- **XMappingMaxPositionError:** Maximum absolute error in that file as shown in the tables below.
- **YMappingFileName:** Name of the mapping file.
- **YMappingLineNumber:** Total number of lines of that file.
- **YMappingColumnNumber:** Total number of columns of that file.
- **YMappingMaxPositionError:** Maximum absolute error in that file must be larger than any entry in the mapping file. To be read properly, the error entries must be in index format, see example.

The X(Y)MappingLineNumber, X(Y)MappingColumnNumber and X(Y)MappingMaxPositionError are only used to check for the correctness of the mapping file.

Example

The following shows an example of the X and Y mapping files:

Matrix X: XYMapping_X.txt

0	-3.00	-2.00	-1.00	0.00	1.00	2.00	3.00
-3.00	-0.00192	-0.00534	-0.00254	0.00023	0.00254	0.00534	0.00192
-2.00	-0.00453	-0.00322	-0.00676	0.00049	0.00676	0.00322	0.00453
-1.00	-0.00331	-0.00845	-0.00769	0.00102	0.00769	0.00845	0.00331
0.00	-0.00787	-0.00228	-0.00787	0	0.00787	0.00228	0.00787
1.00	-0.00232	-0.00210	-0.00342	0.00089	0.00342	0.00210	0.00232
2.00	-0.00134	-0.00308	-0.00675	0.00101	0.00675	0.00308	0.00134
3.00	-0.00789	-0.00148	-0.00234	0.00121	0.00234	0.00148	0.00789

Matrix Y: XYMapping_Y.txt

0	-3.00	-2.00	-1.00	0.00	1.00	2.00	3.00
-3.00	-0.00172	-0.00434	-0.00154	0.00013	0.00204	0.00234	0.00122
-2.00	-0.00433	-0.00222	-0.00376	0.00029	0.00636	0.00222	0.00353
-1.00	-0.00311	-0.00635	-0.00569	0.00089	0.00739	0.00245	0.00231
0.00	-0.00737	-0.00128	-0.00387	0	0.00567	0.00128	0.00387
1.00	-0.00212	-0.00110	-0.00142	0.00079	0.00332	0.00310	0.00132
2.00	-0.00114	-0.00208	-0.00375	0.00089	0.00375	0.00348	0.00122
3.00	-0.00689	-0.00128	-0.00134	0.00101	0.00232	0.00138	0.00689

Verify in the stages.ini for both stages:

```
;--- Travels
MinimumTargetPosition =-3          ; unit
HomePreset =0; unit
MaximumTargetPosition =3           ; unit
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE

The limit travels must be equal or within the X and Y limit positions of the mapping files, +3 and –3, respectively in this example.

Apply the following settings in the system.ini file:

```
;--- Mapping XY
XMappingFileName = XYMapping_X.txt
XMappingLineNumber = 7
XMappingColumnNumber = 7
XMappingMaxPositionError = 0.00845

YMappingFileName = XYMapping_Y.txt
YMappingLineNumber = 7
YMappingColumnNumber = 7
YMappingMaxPositionError = 0.00739
```

Use of the functions:

- **GroupInitialize(XY)**
- **GroupHomeSearch(XY)**
- **GroupMoveAbsolute(XY, 3, 2)**

The mapping files must at least cover the minimum and the maximum travel of the XY group (they must cover the MinimumTargetPosition and the MaximumTargetPosition for the X and Y positioners, parameters defined in the stages.ini, see section Travels). So in the above example, the travel of the X and Y positioners can not be larger than ±3 units, but they can be smaller than this. The units for the data are the same as defined by the EncoderResolution in the stages.ini. The data reads as follows: at position X = 3.00 units, Y = 2.00 units the corrected X position is 2.99852 units (3.00 - 0.00148) and the corrected Y position is 1.99862 units (2.00 - 0.00138). Between two data points, the XPS controller performs a linear interpolation of the error. The two mapping files don't need to contain the same X and Y positions.

NOTE

Mapping is a function implemented within the XPS controller to correct positioning errors. When mapping is activated, it is transparent to the user. At position (X,Y) = (3.00, 2.00), the function GroupPositionCurrentGet(XY.X) doesn't return 2.99852 (3.00 - 0.00148) but 3.

10.5 XYZ Mapping

XYZ mapping is available only with XYZ groups. It compensates for all errors of an XYZ group at any position of that XYZ group. XYZ mapping can be used in conjunction with other compensations, including positioner mapping. Care must be taken to consider the effects when using XYZ mapping and other compensations at the same time.

XYZ mapping is defined by 3 compensation files (compensation for errors in X, Y or Z), in text format. Each of these files can be seen as the juxtaposition of successive tables where the first column of the first table contains the X positions; the first row of the first table contains the Y positions; and the first cell of each table contains one of the Z positions. Each table represents a plane defined by the Z position of the first cell. The separator between the different data in each row is a tab. For legibility, inserting an empty line between successive tables is recommended, but not mandatory. The other cells contain the corresponding error.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

All positions are relative to the physical home position of the XYZ group. The data files must contain the X position = 0, the Y position = 0, and the Z position = 0. The error at X = Y = Z = 0 must be 0, which means that the error at the home position is 0. This hardware reference for XYZ mapping has the advantage of being independent of the value of the HomePreset.

Figure 39 shows the structure for the three mapping files for X, Y, and Z corrections:

- **XYZMappingCorrectionX.dat:** All Err entries are X errors (corrections for X).
- **XYZMappingCorrectionY.dat:** All Err entries are Y errors (corrections for Y).
- **XYZMappingCorrectionZ.dat:** All Err entries are Z errors (corrections for Z).

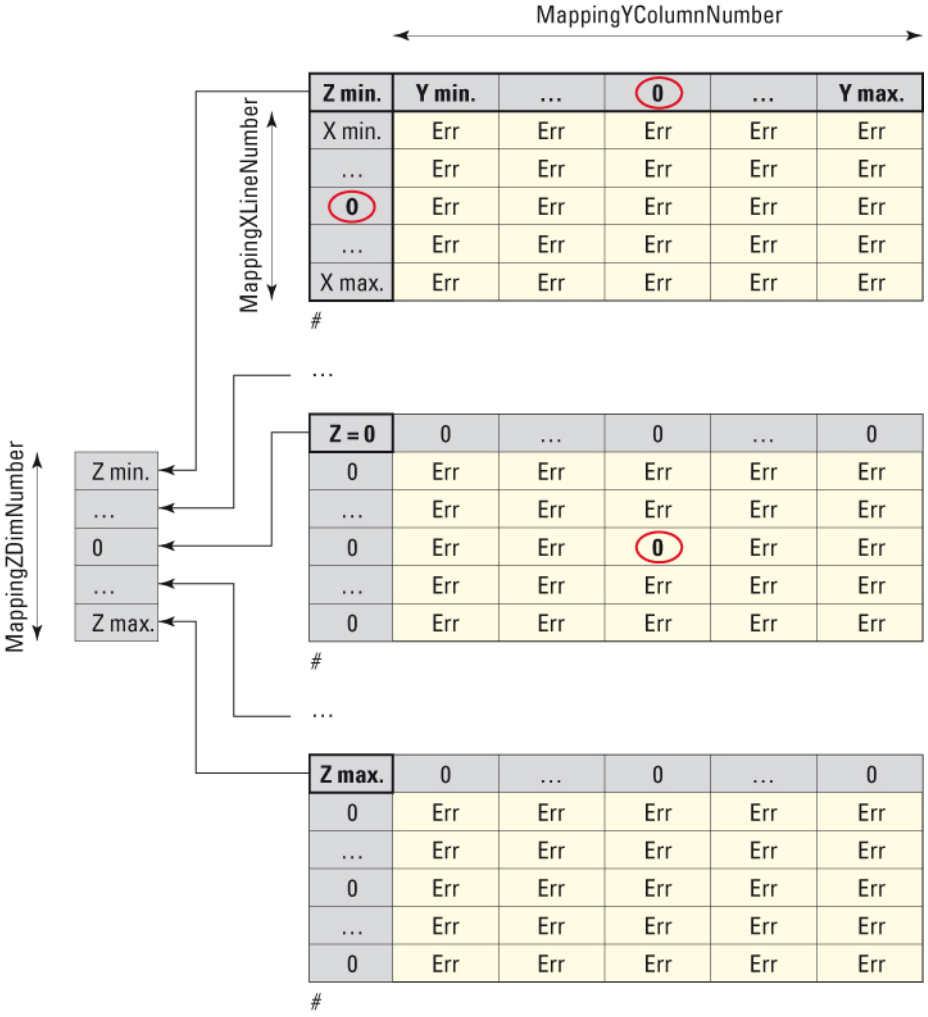


Figure 39: XYZ Mapping Files.
Err in each compensation file can either be Xerr, Yerr or Zerr.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE

The error at X = Y = Z = 0 must be 0. This value in the file corresponds to the HomePreset positions in the XY group reference. A terminator (#) must be added at end of each table.

To activate XYZ mapping, the mapping files must be in the ..\admin\config directory of the XPS controller and the following settings must be configured in the system.ini:

- **XMappingFileName:** Name of the mapping file.
- **XMappingXLineNumber:** Total number of lines of each table including the header.
- **XMappingYColumnNumber:** Total number of columns.
- **XMappingZDimNumber:** Number of tables.
- **XMappingMaxPositionError:** Maximum absolute error in that file must be larger than any entry in the mapping file.
- **YMappingFileName:** Name of the mapping file.
- **YMappingXLineNumber:** Total number of lines of each table including header.
- **YMappingYColumnNumber:** Total number of columns.
- **YMappingZDimNumber:** Number of tables.
- **YMappingMaxPositionError:** Maximum absolute error in that file must be larger than any entry in the mapping file.
- **ZMappingFileName:** Name of the mapping file.
- **ZMappingXLineNumber:** Total number of lines of each table including header.
- **ZMappingYColumnNumber:** Total number of columns.
- **ZMappingZDimNumber:** Number of tables.
- **ZMappingMaxPositionError:** Maximum absolute error in that file must be larger than any entry in the mapping file.

The X(Y,Z)MappingXLineNumber, X(Y,Z)MappingYColumnNumber, X(Y,Z)MappingZDimNumber and X(Y,Z)MappingMaxPositionError are only used to check for the correctness of the mapping file.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

The following example shows the X error mapping files for an XYZ mapping. Note that it is not necessary to repeat the XY coordinates in the table, Z = -1 to the other tables, Z = 0 and Z = 1.

Matrix of X errors: XYZMapping_X.txt

```
 -1.00     -3.00     -2.00     -1.00      0.00      1.00      2.00      3.00
 -3.00   -0.00192  -0.00534   0.00254   0.00125  -0.00137   0.00110   0.00123
 -2.00    0.00453  -0.00322   0.00376  -0.00412  -0.00258  -0.00111  -0.00287
 -1.00   -0.00331   0.00445  -0.00769  -0.00126  -0.00153   0.00298   0.00487
  0.00   -0.00787   0.00228  -0.00787   0.00320   0.00154  -0.00169  -0.00369
  1.00    0.00232   0.00210  -0.00342   0.00169   0.00265   0.00169   0.00125
  2.00   -0.00134   0.00308   0.00275  -0.00369   0.00337  -0.00214  -0.00456
  3.00    0.00189  -0.00148   0.00234   0.00458  -0.00333   0.00152   0.00335
#
  0         0         0         0         0         0         0         0
  0       -0.00192  -0.00534   0.00254   0.00125  -0.00137   0.00110   0.00123
  0        0.00453  -0.00322   0.00376  -0.00412  -0.00258  -0.00111  -0.00287
  0       -0.00331   0.00445  -0.00769  -0.00126  -0.00153   0.00298   0.00487
  0       -0.00787   0.00228  -0.00787   0         0.00154  -0.00169  -0.00369
  0        0.00232   0.00210  -0.00342   0.00169   0.00265   0.00169   0.00125
  0       -0.00134   0.00308   0.00275  -0.00369   0.00337  -0.00214  -0.00456
  0        0.00189  -0.00148   0.00234   0.00458  -0.00333   0.00152   0.00335
#
  1.00      0         0         0         0         0         0         0
  0       -0.00192  -0.00534   0.00254   0.00125  -0.00137   0.00110   0.00123
  0        0.00453  -0.00322   0.00376  -0.00412  -0.00258  -0.00111  -0.00287
  0       -0.00331   0.00445  -0.00769  -0.00126  -0.00153   0.00298   0.00487
  0       -0.00787   0.00228  -0.00787   0.00320   0.00154  -0.00169  -0.00369
  0        0.00232   0.00210  -0.00342   0.00169   0.00265   0.00169   0.00125
  0       -0.00134   0.00308   0.00275  -0.00369   0.00337  -0.00214  -0.00456
  0        0.00189  -0.00148   0.00234   0.00458  -0.00333   0.00152   0.00335
#
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Matrix of Y errors: XYZMapping_Y.txt

```
  -1.00      -3.00      -2.00      -1.00      0.00       1.00       2.00       3.00
  -3.00      -0.00190   -0.00530   0.00190    0.00125    -0.00190   0.00530    0.00190
  -2.00      -0.00190   -0.00530   0.00190    0.00125    -0.00190   0.00530    0.00190
  -1.00      -0.00190   -0.00530   0.00190    0.00125    -0.00190   0.00530    0.00190
   0.00      -0.00190   -0.00530   0.00190    0.00125    -0.00190   0.00530    0.00190
   1.00      -0.00190   -0.00530   0.00190    0.00125    -0.00190   0.00530    0.00190
   2.00      -0.00190   -0.00530   0.00190    0.00125    -0.00190   0.00530    0.00190
   3.00      -0.00190   -0.00530   0.00190    0.00125    -0.00190   0.00530    0.00190
#
   0          0          0          0          0          0          0          0
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0          -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
#
   1.00       0          0          0          0          0          0          0
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
   0          -0.00192   -0.00534   0.00254    0.00125    -0.00137   0.00110    0.00123
#
```

Matrix of Z errors: XYZMapping_Z.txt

```
  -1.00      -3.00      -2.00      -1.00      0.00       1.00       2.00       3.00
  -3.00      -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
  -2.00      -0.0003    -0.0003    0.0003     0.0003     -0.0003    -0.0003    0.0003
  -1.00      -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
   0.00      -0.0003    -0.0003    0.0003     0.0003     -0.0003    -0.0003    0.0003
   1.00      -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
   2.00      -0.0003    -0.0003    0.0003     0.0003     -0.0003    -0.0003    0.0003
   3.00      -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
#
   0          0          0          0          0          0          0          0
   0          -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
   0          -0.0003    -0.0003    0.0003     0.0003     -0.0003    -0.0003    0.0003
   0          -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
   0          -0.0003    -0.0003    0.0003     0          -0.0003    -0.0003    0.0003
   0          -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
   0          -0.0003    -0.0003    0.0003     0.0003     -0.0003    -0.0003    0.0003
   0          -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
#
   1.00       0          0          0          0          0          0          0
   0          -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
   0          -0.0003    -0.0003    0.0003     0.0003     -0.0003    -0.0003    0.0003
   0          -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
   0          -0.0003    -0.0003    0.0003     0.0003     -0.0003    -0.0003    0.0003
   0          -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
   0          -0.0003    -0.0003    0.0003     0.0003     -0.0003    -0.0003    0.0003
   0          -0.0002    -0.0002    0.0002     0.0002     -0.0002    -0.0002    0.0002
#
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Verify in the corresponding sections of the stages.ini:

For the X axis:

```
;--- Travels
MinimumTargetPosition =-3          ; unit
HomePreset =0; unit
MaximumTargetPosition =3           ; unit
```

NOTE

The limit travels must be equal or within the X limit positions of the mapping files (shown here +3 and -3).

For the Y axis:

```
;--- Travels
MinimumTargetPosition =-3          ; unit
HomePreset =0; unit
MaximumTargetPosition =3           ; unit
```

NOTE

The limit travels must be equal or within the Y limit positions of the mapping files (shown here +3 and -3).

For Z axis:

```
;--- Travels
MinimumTargetPosition =-1          ; unit
HomePreset =0; unit
MaximumTargetPosition =1           ; unit
```

NOTE

The limit travels must be equal or within the Z limit positions of the mapping files (shown here +1 and -1).

In the system.ini file:

```
;--- Mapping XYZ
XMappingFileName = XYZMapping_X.txt
XMappingXLineNumber = 7
XMappingYColumnNumber = 7
XMappingZDimNumber = 3
XMappingMaxPositionError = 0.00787
YMappingFileName = XYZMapping_Y.txt
YMappingXLineNumber = 7
YMappingYColumnNumber = 7
YMappingZDimNumber = 3
YMappingMaxPositionError = 0.00534
ZMappingFileName = XYZMapping_Z.txt
ZMappingXLineNumber = 7
ZMappingYColumnNumber = 7
ZMappingZDimNumber = 3
ZMappingMaxPositionError = 0.0003
```

Represents the errors in the X axis.

Represents the errors in the Y axis.

Represents the errors in the Z axis.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Use of the functions:

- **GroupInitialize(XYZ)**
- **GroupHomeSearch(XYZ)**
- **GroupMoveAbsolute(XYZ, 3, 1, 1)**

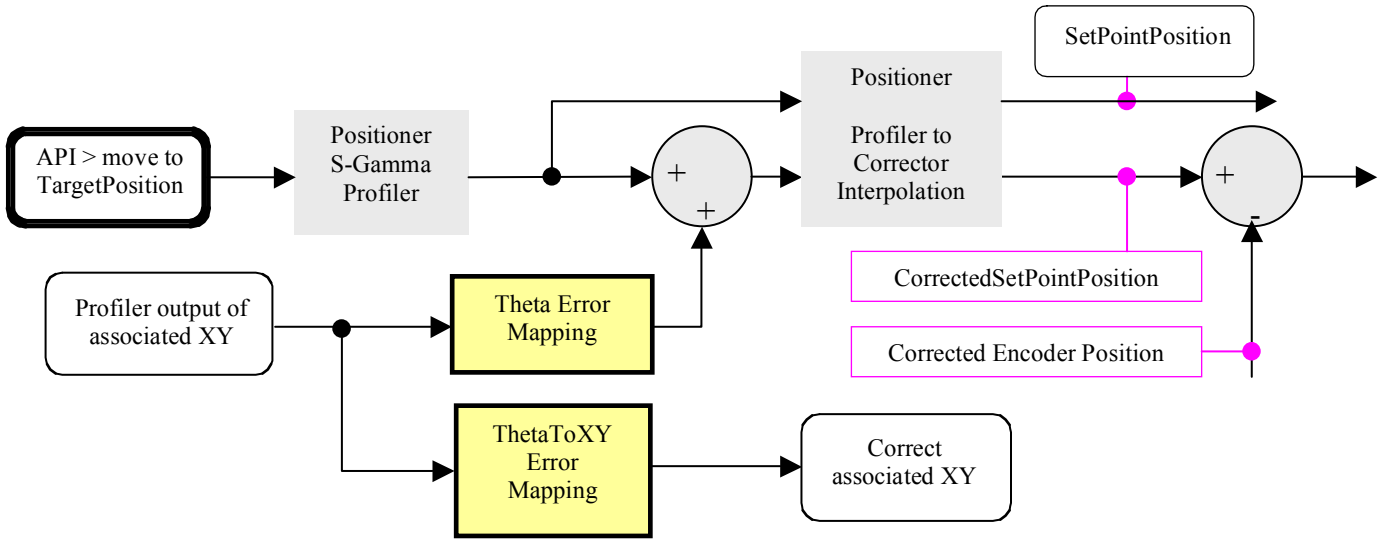
The mapping files must at least cover the minimum and the maximum travel of the XYZ group (they must cover the MinimumTargetPosition and the MaximumTargetPosition for the X, Y and Z positioners, parameters defined in the stages.ini, see section Travels). So in the above example the travel of the X and Y positioners can not be larger than ±3 units, and the travel for the Z positioner can not be larger than ±1 unit. But the travel can be smaller than these. The unit of the data is the same as defined by EncoderResolution in the stages.ini. The data reads as follows: at position (X,Y,Z) = (3.00, 2.00, 1.00), the corrected X position is 2.99848 units (3.00 - 0.00152), the corrected Y position is 2.9989 units (3.00 - 0.00110) and the corrected Z position is 3.0002 units (3.00 + 0.0002). Between two datas, the XPS controller does a linear interpolation of the error. The three mapping files for X, Y, and Z don't need to contain the same X, Y and Z positions.

NOTE

Mapping is a function implemented in the XPS controller to correct errors. But when mapping is activated, it is transparent to the user. At position (X,Y,Z) = (3.00, 1.00, 1.00), the function GroupPositionCurrentGet(XYZ.X) doesn't return 3.00333 (3.00 + 0.00333) but returns 3.

10.6 "Yaw" Mapping (PP Firmware Version Only)

During an XY move, due to surface flatness errors in the base, the Theta axis (compared with the base) turns slightly around its center. So a correction, DeltaTheta value, calculated from the XY positions, must be sent to Theta to move it in order to keep the Theta immobile relative the base, during the XY move. In the following discussion, "Theta" is a SingleAxisTheta Group.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Configuration in the system.ini file:

Yaw mapping is enabled when an "XY" group is associated with the "Theta" group. It is defined by this key word:

```
YawMappingXYGroupName
```

The XY motions induce errors along "Theta". These errors are defined in a mapping file:

```
YawMappingToThetaFileName
```

When the Yaw is applied, corrections are generated on XY. These corrections are defined in the two mapping files:

```
YawMappingToXFileName
YawMappingToYFileName
```

In building the mapping file YawMappingToThetaMappingFile, follow the same rules used for positioner mapping. The following parameters are used to check the correctness of the file:

```
YawMappingToThetaLineNumber
YawMappingToThetaColumnNumber
YawMappingToThetaMaxPositionError
```

In building the two mapping files YawMappingToXMappingFile and YawMappingToYMappingFile, follow the same rules used for XY group mapping. The following parameters are used to check for file correctness:

```
YawMappingToXFileName
YawMappingToXLineNumber
YawMappingToXColumnNumber
YawMappingToXMaxPositionError
YawMappingToYFileName
YawMappingToYLineNumber
YawMappingToYColumnNumber
YawMappingToYMaxPositionError
```

Format for each yaw mapping file (X, Y and Theta):

- First cell must be 0.
- First column represents the X Positions.
- First row represents the Y Positions.
- Each cell represents the corresponding yaw error

NOTE

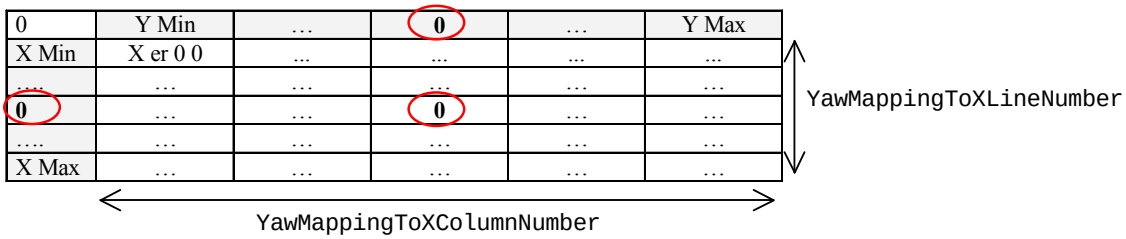
The error of X = Y = 0 must be 0 (This XY position in the file correspond to the HomePreset position of the XY group reference). X and Y positions must at least cover the entire travel of the XY group. XY and Yaw corrections occur at the same time.

Limitations of mapping table size:

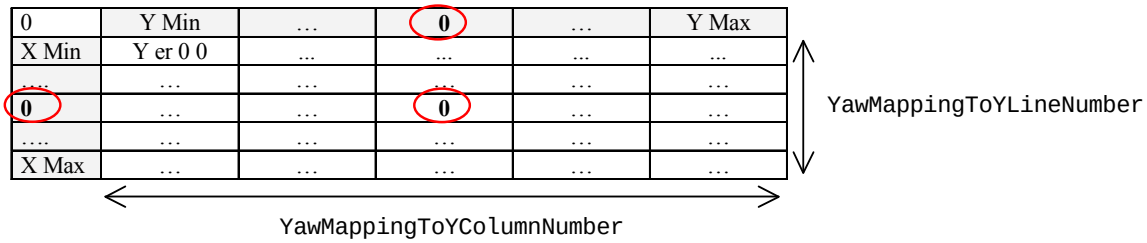
- Maximum lines = **400**
- Maximum columns = **400**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

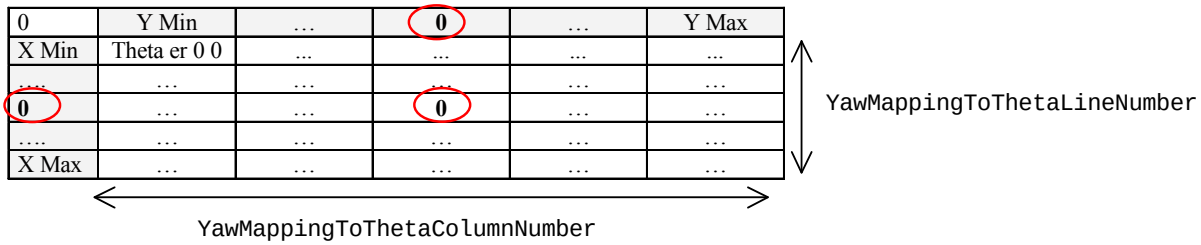
Format of the "YawMappingToXFileName" file :



Format of the "YawMappingToYFileName" file :



Format of the "YawMappingToThetaFileName" file :

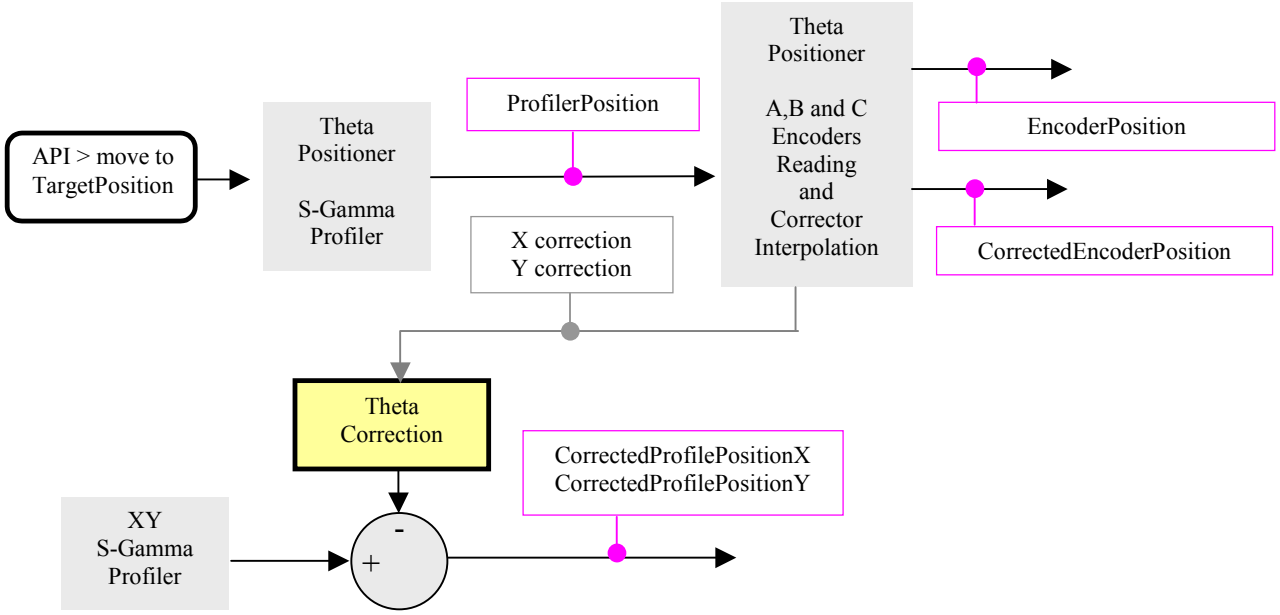


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10.7 "Theta" Encoder and XY Correction

In a Theta-XY group, a motion in Theta will induce an offset of the center of the Theta-axis. Utilizing the 3 encoders of the Theta stage, a correction in X and Y can be implemented to correct for the induced eccentricity, effectively keeping the Theta axis in the same position relative to the base. The "Theta" axis is composed of three encoders A, B and C. The encoder type is defined in the **stages.ini** file by "**EncoderType"**. It must either be "AquadBTheta" or "AnalogInterpolatedTheta".

```
EncoderType =      ; AquadBTheta or AnalogInterpolatedTheta
```

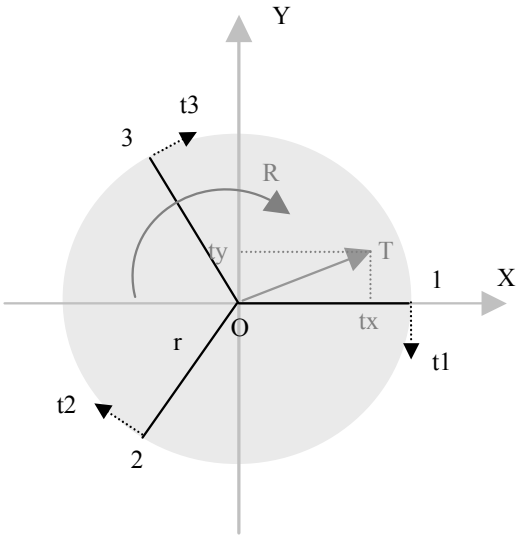


To enable "Theta correction", an XY group must be associated with the **SingleAxisTheta** group. It is defined in the **system.ini** file by:

```
ThetaCorrectionXYGroupName =
ThetaCorrectionLowPassCutOffFrequency = 20        ; Hz
```

The radius "r" and the XY correction limits are defined in the **stages.ini** file by:

```
EncoderRadius      =          ; units XY * rad / units Theta
MaximumEncoderCorrectionX  =      ; units XY
MaximumEncoderCorrectionY  =      ; units XY
```



O : Center
1,2,3 : Encoders
r : EncoderRadius
t1, t2, t3 : delta encoder counter

T1 : Encoder position 1
T2 : Encoder position 2
T3 : Encoder position 3

R : Rotation $R = \frac{T1 + T2 + T3}{3}$

T : Translation $\begin{pmatrix} tx \\ ty \end{pmatrix} = \begin{pmatrix} \frac{T3 - T2}{\sqrt{3}}.r \\ \left(\frac{(T1 + T2 + T3)}{3} - T1 \right).r \end{pmatrix}$ *X correction* *Y correction*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

11.0 Event Triggers

XPS event triggers work similar to IF/THEN statements in programming. “If” the **event** occurs, “then” an **action** is triggered. Programmers can trigger any action (from a list of possible actions, see section 11.2) at any event (from a large list of possible events, see section 11.1). It is also possible to trigger several actions with the same event. Furthermore, it is possible to link several events to an event configuration. In this case, all events must happen at the same time to trigger the action(s). It is comparable to a logic AND between the different events.

Some events are one-time events like “motion start”. They will trigger an action only once when the event occurs. Some other events have a duration like “motion state”. They will trigger the same action each time (as applicable) as long as the event occurs. For events with duration, the event can be also considered as a statement that is checked whether it is true or not. A third event category are the permanent events “Always” (always happens) and “Timer” (happens every nth servo cycle). They will trigger the action always on every nth servo cycle.

As the XPS controller provides the utmost flexibility in programming event triggers, the user must be careful and consider possible unwanted effects. Some events might have a duration although only one single action is asked. Some other events might never occur. This is especially true when linking several events to an event configuration. The different possible effects are illustrated in section 11.3 by a few examples.

To trigger an action with an event, the event and the associated action must first be configured using the functions **EventExtendedConfigurationTriggerSet()** and **EventExtendedConfigurationActionSet()**. Then, the event trigger is activated using the function **EventExtendedStart()**. When activated, the XPS controller checks for the event at each servo cycle (or each profiler cycle for those events that are motion related) and triggers the action when the event happens. Hence, the maximum latency between the event and the action is equal to the servo cycle of 100 µs or equal to the profiler cycle time of 400 µs. For events with duration, it means that the same action is triggered at each servo cycle, i.e. every 100 µs, or at each profiler cycle, i.e. every 400 µs, as long as the event is happening.

Event triggers (and their associated actions) are automatically removed after the event configuration has happened at least once and is no longer true. The only exception is if the event configuration contains any of the permanent events “Always” or “Timer”. In this case the event trigger will always stay active. With the function **EventExtendedRemove()**, any event trigger can get removed.

The function **EventExtendedWait()** can be used to halt a process. It essentially blocks the socket until the event occurs. Once the event occurs, it is deleted. It requires a preceding function **EventExtendedConfigurationTriggerSet()** to define the event at which the process continues.

The functions **EventExtendedGet()** and **EventExtendedAllGet()** return details of the event and action configurations.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

11.1 Events

General events are defined as "**Always**", "**Immediate**" and "**Timer**". With the event "Always", an action is triggered each servo cycle, meaning every 100 µs. For events that are defined as "Immediate", an action is triggered once immediately (during the very next servo cycle). For the events defined as "Timer", an action is triggered immediately and every nth servo cycle. Here, "n" corresponds to the "FrequencyTicks" defined in the function **TimerSet()**. There are five different timers available that can be selected by the actor (1…5) (Actor is the object that actions/events are linked to).

All events that are motion related (from MotionStart to TrajectoryPulseOutputState in the below table, except MotionDone) refer to the motion profiler of the XPS controller. The motion profiler runs at a frequency of 2.5 kHz, or every 400 µs. Thus, events triggered by the motion profiler have a resolution of 400 µs. Consequently, events with duration, such as MotionState, will trigger an action every 400 µs. All motion related events, except MotionDone, have a category such as "Sgamma" or "Jog". This category refers to the motion profiler. Here, SGamma refers to the profiler used with the function GroupMoveRelative and GroupMoveAbsolute and Jog refers to the profiler used in the Jogging state. The other event categories refer to trajectories. The separator between the category, the actor, and the event name is a dot (.).

Actor								Category							Event Name	Parameter			
Group			GPIO					SGamma			XYLineArc			PVT		1	2	3	4
Positioner			TimerX					Jog			Spline								
															Immediate				
															Always				
							■								Timer				
						■			■	■					MotionStart				
						■			■	■					MotionStop				
						■			■	■					MotionState				
						■			■	■					ConstantVelocityStart				
						■			■						ConstantVelocityEnd				
						■			■	■					ConstantVelocityState				
						■			■						ConstantAccelerationStart				
						■			■						ConstantAccelerationEnd				
						■			■						ConstantAccelerationState				
						■			■						ConstantDecelerationStart				
						■			■						ConstantDecelerationEnd				
						■			■						ConstantDecelerationState				
					■						■	■	■		TrajectoryStart				
					■						■	■	■		TrajectoryEnd				
					■						■	■	■		TrajectoryState				
					■						■	■	■		ElementNumberStart	Element #			
					■						■	■	■		ElementNumberState	Element #			
						■									MotionDone				
					■						■		■		TrajectoryPulse				
					■						■		■		TrajectoryPulseOutputState				
				■											DILowHigh	Bit index			
				■											DIHighLow	Bit index			
				■											DIToggled	Bit index			
				■											ADCHighLimit	Value			
				■											ADCLowLimit	Value			
			■												PositionerError	Mask			
			■												PositionerHardwareStatus	Mask			

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

An event is entirely composed of:

[Actor].[Category].Event Name, Parameter1, Parameter2, Parameter3, Parameter4

Not all event names have a preceding actor and category, but all events have four parameters, even though some parameters are not needed. For these parameters, it is still required to use zero (0) as default.

To define an Event, use the function EventExtendedConfigurationTriggerSet().

Examples

EventExtendedConfigurationTriggerSet (MyGroup.MyPositioner.SGamma.MotionStart, 0, 0, 0, 0)

In this case, the actor is a positioner (MyGroup.MyPositioner) and the event has a category. The event happens when the next motion with the SGamma profiler on the positioner MyGroup.MyPositioner starts. After the motion has started, the event is removed.

EventExtendedConfigurationTriggerSet (MyGroup.XYLineArc.ElementNumberStart, 5, 0, 0, 0)

In this case, the actor is a group (MyGroup) and the event has a category. The event happens when the trajectory element number 5 on the next LineArc trajectory on this group starts.

EventExtendedConfigurationTriggerSet (GPIO2.ADC2.ADCHighLimit, 3, 0, 0, 0)

In this case, the actor is a GPIO name (GPIO2.ADC2) and the event has no category. The event happens when the voltage on the GPIO.ADC2 exceeds 3 Volts.

It is also possible to link different events to an event configuration. The same function EventExtendedConfigurationTriggerSet() is used, and the different events are just separated by a comma. The event combination happens when all individual events happen at the same time. It is comparable to a logic AND between the different events.

Examples

EventExtendedConfigurationTriggerSet (GPIO2.ADC2.ADCHighLimit, 3, 0, 0, 0, MyGroup.MyPositioner.SGamma.MotionState, 0, 0, 0, 0)

This event will happen when the voltage of the GPIO.ADC2 exceeds 3 Volts during a SGamma motion of the MyGroup.MyPositioner.

EventExtendedConfigurationTriggerSet (Always, 0, 0, 0, 0, MyGroup.MyPositioner.SGamma.MotionStart, 0, 0, 0, 0)

This event will happen during each SGamma motion starts of the positioner MyGroup.MyPositioner. The addition of the event Always has the effect of keeping the event after the next motion has been started (see differences compared to the first example above).

The exact meaning of the different events and event parameters are as follows:

Always:	Triggers an action ALWAYS, means at each servo cycle. Event parameter 1 to 4 = 0 by default. **NOTE:** This event is PERMANENT until the next reboot. Call the EventExtendedRemove function to remove it.
Immediate:	Triggers an action IMMEDIATELY, meaning once during the very next servo cycle: Event parameter 1 to 4 = 0 by default. **NOTE:** This event is TEMPORARY.
Timer:	Triggers an action every nth servo cycle, where n is defined with the function TimerSet. Event parameter 1 to 4 = 0 by default.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE: This event is PERMANENT until the next reboot. Call the EventExtendedRemove function to remove it.

MotionDone: Triggers an action when a position is reached.

Event parameter 1 to 4 = 0 by default.

For the exact definition of MotionDone, please refer to section 7.5.

ConstantVelocityStart: Triggers an action when constant velocity is reached. Event parameter 1 to 4 = 0 by default.

ConstantVelocityEnd: Triggers an action when constant velocity is finished. Event parameter 1 to 4 = 0 by default.

ConstantVelocityState: Triggers an action during constant velocity. Event parameter 1 to 4 = 0 by default.

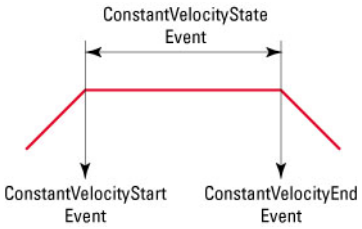


Figure 40: Constant Velocity Event.

ConstantAccelerationStart: Triggers an action when constant acceleration is reached. Event parameter 1 to 4 = 0 by default.

ConstantAccelerationEnd: Triggers an action when constant acceleration is finished. Event parameter 1 to 4 = 0 by default.

ConstantAccelerationState: Triggers an action during constant acceleration. Event parameter 1 to 4 = 0 by default.

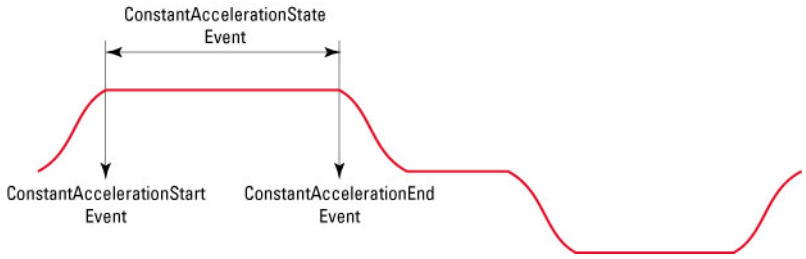


Figure 41: Constant Acceleration Event.

The same definition applies to ConstantDecelerationStart, ConstantDecelerationEnd and ConstantDecelerationState.

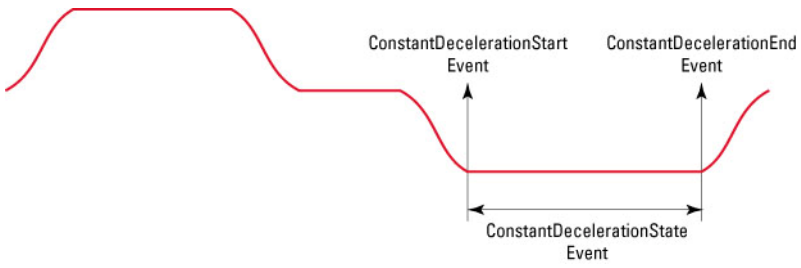


Figure 42: Constant Deceleration Event.

MotionStart: Triggers an action when motion starts. Event parameter 1 to 4 = 0 by default.

MotionEnd: Trigger an action when motion is ended. Event parameter 1 to 4 = 0 by default. Note, MotionEnd refers to the end of

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

the theoretical motion which is not the same as the definition of MotionDone (see section 7.5).

MotionState: Triggers an action during motion. Event parameter 1 to 4 = 0 by default.

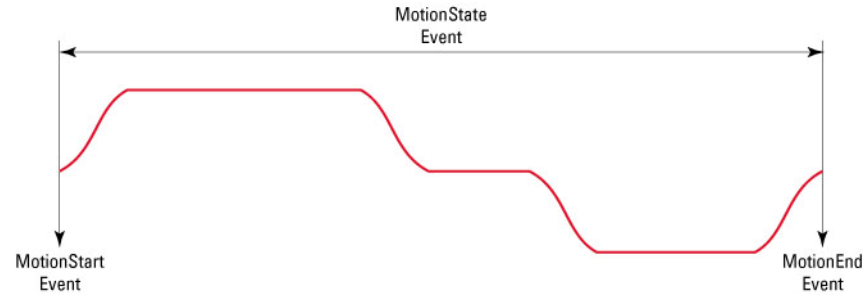


Figure 43: Motion Event.

There are also several trajectory events that can be defined:

TrajectoryStart: Triggers an action when the trajectory has started. Event parameter 1 to 4 = 0 by default.

TrajectoryEnd: Triggers an action when the trajectory has stopped. Event parameter 1 to 4 = 0 by default.

TrajectoryState: Triggers an action during trajectory execution. Event parameter 1 to 4 = 0 by default.

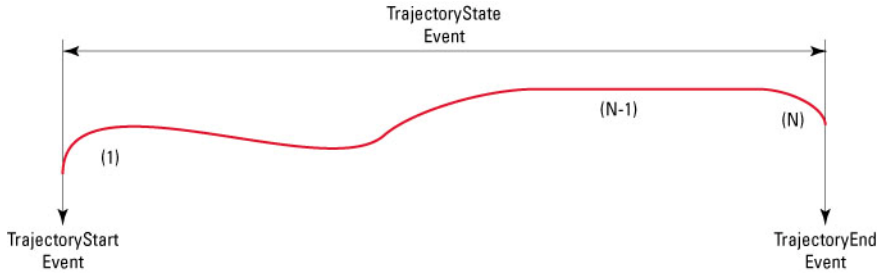


Figure 44: Trajectory Event.

ElementNumberStart: Triggers an action when the trajectory element number has started. The first event parameter specifies the number of the trajectory element. The other event parameters are 0 by default.

ElementNumberState: Triggers an action during the execution of that trajectory element number. The first event parameter specifies the number of the trajectory element. The other event parameters are 0 by default.

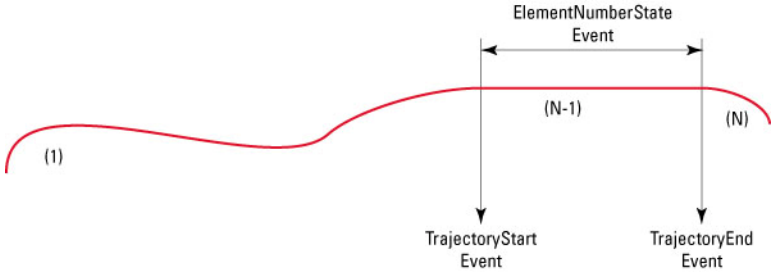


Figure 45: Element Number Event.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TrajectoryPulse: Triggers an action when a pulse on the trajectory is generated (see chapter 13.0: ““Output Triggers for details). All event parameters are 0 by default.

TrajectoryPulseOutputState: Triggers an action during the trajectory pulse output state, meaning between the start and the end of the trajectory output pulses (see sections **Erreur ! Source du renvoi introuvable.** and **Erreur ! Source du renvoi introuvable.**: “Triggers on Trajectories” for details). All event parameters are 0 by default.

ILowState: Triggers an action when the digital input bit is in a low state. The first event parameter is the bit index (0 to 15). The other event parameters are 0 by default.

DILowHigh: Triggers an action when the digital input bit switches from a low state to a high state. The first event parameter is the bit index (0 to 15). The other event parameters are 0 by default.

DIHighState: Triggers an action when the digital input bit is in a high state. The first event parameter is the bit index (0 to 15). The other event parameters are 0 by default.

DIHighLow: Triggers an action when the digital input bit switches from a high to a low state. The first event parameter is the bit index (0 to 15). The other event parameters are 0 by default.

DIToggled: Triggers an action when the digital input bit switches from low to high or from high to low. The first event parameter is the bit index (0 to 15). The other event parameters are 0 by default.

ADCHighLimit: Triggers an action when the analog input value exceeds the limit. The first event parameter is the limit value in volts. The other event parameters are 0 by default.

ADCLowLimit: Triggers an action when the analog input value is below the limit. The first event parameter is the limit value in volts. The other event parameters are 0 by default.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PositionerError: Triggers an action when the current positioner error applied with the error mask (for the 32 bit register) results in a value other than zero. The first event parameter specifies the error mask in a decimal format. The other event parameters are 0 by default.

Code (Hexa)	Bit #	Decimal	Positioner error description
0			No error
0x00000001	0	1	General inhibition detected
0x00000002	1	2	Fatal following error detected
0x00000004	2	4	Home search time out
0x00000008	3	8	Motion done time out
0x00000010	4	16	Requested position exceed travel limits in trajectory or slave mode
0x00000020	5	32	Requested velocity exceed maximum value in trajectory or slave mode
0x00000040	6	64	Requested acceleration exceed max value in trajectory or slave mode
0x00000100	8	256	Minus end of course activated
0x00000200	9	512	Plus end of course activated
0x00000400	10	1024	Minus end of run glitch
0x00000800	11	2048	Plus end of run glitch
0x00001000	12	4096	Encoder quadrature error
0x00002000	13	8192	Encoder frequency and coherence error
0x00010000	16	65536	Hard interpolator encoder error
0x00020000	17	131072	Hard interpolator encoder quadrature error
0x00100000	20	1048576	First driver in fault
0x00200000	21	2097152	Second driver in fault

Examples

EventExtendedConfigurationTriggerSet (MyGroup.MyPositioner.PositionerError, 2, 0, 0, 0)

This event happens when the positioner MyGroup.MyPositioner has a fatal following error.

EventExtendedConfigurationTriggerSet (MyGroup.MyPositioner.PositionerError, 12, 0, 0, 0)

This event happens when the positioner MyGroup.MyPositioner has either a home search time out or a motion done time out.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PositionerHardwareStatus: Triggers an action when the current hardware status applied with the error mask results in a value other than zero. The first event parameter specifies the status mask in decimal format. The other event parameters are 0 by default.

Code (Hexa)	Bit #	Decimal	Hardware status description
0x00000001	0	1	General inhibition detected
0x00000004	2	4	ZM high level
0x00000100	8	256	Minus end of run activated
0x00000200	9	512	Plus end of run activated
0x00000400	10	1024	Minus end of run glitch
0x00000800	11	2048	Plus end of run glitch
0x00001000	12	4096	Encoder quadrature error
0x00002000	13	8192	Encoder frequency or coherence error
0x00010000	16	65536	Hard interpolator encoder error
0x00020000	17	131072	Hard interpolator encoder quadrature error
0x00100000	20	1048576	First driver in fault
0x00200000	21	2097152	Second driver in fault
0x00400000	22	4194304	First driver powered on
0x00800000	23	8388608	Second driver powered on

Example

EventExtendedConfigurationTriggerSet
(MyGroup.MyPositioner.PositionerHardwareStatus, 768, 0, 0, 0)

This event happens when the positioner MyGroup.MyPositioner either the plus end of run or a minus end of run is detected.

WarningFollowingError: Triggers an action when the following error exceeds the warning following error value. In the PositionCompare mode (activated by the PositionerPositionCompareEnable() function), during a move (relative or absolute) and inside the zone set by PositionerPositionCompareSet(), if the current following error exceeds the WarningFollowingError value, the PositionCompareWarningFollowingErrorFlag is activated and the move returns a corresponding error (-120 : Warning following error during move with position compare enabled).

To reset the PositionCompareWarningFollowingErrorFlag, send the PositionerPositionCompareDisable() function.

The WarningFollowingError is set to FatalFollowingError (defined in stages.ini file) by default, but it can be modified with PositionerWarningErrorSet().

Example

EventExtendedConfigurationTriggerSet

(MyGroup.MyPositioner. WarningFollowingError, 0, 0, 0, 0)

This event happens when the positioner MyGroup.MyPositioner has a following error that exceeds the warning following error value.

DoubleGlobalArrayEqual: Triggers an action when the value of the variable in the DoubleGlobalArray and referenced by the global variable number is equal to the value to check. The variable can be modified by using the DoubleGlobalArraySet() function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DoubleGlobalArrayDifferent: Triggers an action when the value of the variable in the DoubleGlobalArray and referenced by the global variable number is different from the value to check. The variable can be modified by using the DoubleGlobalArraySet() function.

DoubleGlobalArrayInferiorOrEqual: Triggers an action when the value of the variable in the DoubleGlobalArray and referenced by the global variable number is less than or equal to the value to check. The variable can be modified by using the DoubleGlobalArraySet() function.

DoubleGlobalArraySuperiorOrEqual: Triggers an action when the value of the variable in the DoubleGlobalArray and referenced by the global variable number is greater than or equal to the value to check. The variable can be modified by using the DoubleGlobalArraySet() function.

DoubleGlobalArrayInferior: Triggers an action when the value of the variable in the DoubleGlobalArray and referenced by the global variable number is lower than the value to check. The variable can be modified by using the DoubleGlobalArraySet() function.

DoubleGlobalArraySuperior: Triggers an action when the value of the variable in the DoubleGlobalArray and referenced by the global variable number is higher than the value to check. The variable can be modified by using the DoubleGlobalArraySet() function.

DoubleGlobalArrayInWindow: Triggers an action when the value of the variable in the DoubleGlobalArray and referenced by the global variable number is superior to MinValue and inferior to MaxValue.

DoubleGlobalArrayOutWindow: Triggers an action when the value of the variable in the DoubleGlobalArray and referenced by the global variable number is outside the interval defined by MinValue and MaxValue.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

11.2 Actions

There are several actions that can be triggered by the events discussed previously. Users have the full flexibility to trigger any action (out of the list of possible actions) at any event (out of the list of possible events). It is also possible to trigger several actions at the same event by adding several sets of parameters to the function **EventExtendedConfigurationActionSet()**, similar to how it is done with events.

Actor			Action Name	Parameter			
Group	GPIO						
Positioner		TimerX		1	2	3	4
	■		DOToggle	Mask			
	■		DOPulse	Mask			
	■		DOSet	Mask	Value		
	■		DACSet.CurrentPosition	Positioner name	Gain	Offset	
	■		DACSet.CurrentVelocity	Positioner name	Gain	Offset	
	■		DACSet.SetpointPosition	Positioner name	Gain	Offset	
	■		DACSet.SetpointVelocity	Positioner name	Gain	Offset	
	■		DACSet.SetpointAcceleration	Positioner name	Gain	Offset	
			ExecuteTCLScript	TCL file name	Task name	Arguments	
			KillTCLScript	Task name			
			GatheringOneData				
			GatheringRun	Nb of points	Divisor		
			GatheringRunAppend				
			GatheringStop				
			ExternalGatheringRun	Nb of points	Divisor		
■			MoveAbort				

CAUTION

Certain events like MotionState have a duration. These events trigger the associated action in each motion profiler cycle as long as the event is true. For example, associating the action DOToggle with the event MotionState will toggle the value of the digital output in each profiler cycle as long as the MotionState event is true.



An event doesn't reset the action after the event: For example, to set a digital output to a certain value during a constant velocity state and to set it to its previous value afterwards, two event triggers are needed: One to set to the digital output of the desired value at the event ConstantVelocityStart and another one to set it to its original value at the event ConstantVelocityEnd. The same effect CANNOT be achieved by using the event ConstantVelocityState by itself.

An action is composed entirely of:

[Actor].Action Name, Parameter1, Parameter2, Parameter3, Parameter4.

Not all action names have a preceding actor, but all actions have four parameters. Even though all four parameters may not be defined in an action, it is still required to have an entry, with zero (0) as the default.

To define an action, use the function **EventExtendedConfigurationActionSet()**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example:

EventExtendedConfigurationActionSet (GPIO1.DO.DOToggled, 4, 0, 0, 0)

In this case the actor is the digital output GPIO1.DO and the action is to toggle the output. The value 4 refers to bit #3, 00000100. Hence, this action toggles the value of bit 3 on the digital output GPIO.DO.

EventExtendedConfigurationActionSet (ExecuteTCLScript, Example.tcl, 1, 0, 0)

The action ExecuteTCLScript has no preceding actor. This action will execute the TCL script "Example.tcl". The task name is 1 and the TCL script has no arguments (a zero for the third parameter means there are no arguments).

EventExtendedConfigurationActionSet (GatheringRun, 1000, 10, 0, 0)

The action GatheringRun has no preceding actor. This action will start an internal data gathering. It will gather a total of 1000 data points, one data point every 10th servo cycle, meaning one data point for every 10/10000 s = 1 ms.

It is also possible to trigger several actions with the same event. To do so, just define another action in the SAME function. Several actions must be separated by a comma (,).

Example:

EventExtendedConfigurationTriggerSet (MyGroup.MyPositioner.PositionerError, 2, 0, 0, 0)

EventExtendedConfigurationActionSet (ExecuteTCLScript, ShutDown.tcl, 1, 0, 0, ExecuteTCLScript, ErrorDiagnostic.tcl, 2, 0, 0)

EventExtendedStart ()

In this example, the TCL scripts ShutDown.tcl and ErrorDiagnostic.tcl are executed when a fatal following error is detected on the positioner MyGroup.MyPositioner.

The exact meaning of the different actions and action parameters is as follows:

DOToggle: This action is used to reverse the value of one or many bits of 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 at each profiler cycle as long as the event occurs.

Action Parameter #1 – 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, bit #3 will be toggled.
Action Parameter #2 to #4	These parameters are 0 by default.

DOPulse: This action is used to generate a positive pulse on the Digital Output. The duration of the pulse is 1 microsecond. To function, 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 µs pulse will be generated at each cycle of the motion profiler (or every 400 µs) as long as the event occurs.

Action Parameter #1 – 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 µs pulse will be generated on bit #2 and #3 of the GPIO output.
Action Parameter #2 to #4	These parameters are 0 by default.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DOSet: This action is used to modify the value of bit(s) on a Digital Output.

Action Parameter #1 – 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 bits # 2, #4 and #5 are being addressed on the GPIO output.
Action Parameter #2 – Value	This parameter sets the value of the bits that are being addressed according to the Mask setting. 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.
Action parameter #3 and #4	These parameters are 0 by default.

DACSet.CurrentPosition and **DACSet.SetpointPosition:** This action sets a voltage on the Analog output in relation to the actual (current) or theoretical (Setpoint) position. The gain and offset are used to calibrate the output. This action makes the most sense with events that have some duration (always, MotionState, ElementNumberState, etc.) as the analog output will be updated at each servo cycle or at each profiler cycle as long as the event occurs. When used with events that have no duration (like MotionStart or MotionEnd), the analog output is only updated once and this value is kept until it is changed.

Action Parameter #1 – Positioner Name	This parameter defines the name of the positioner on which the position value is used.
Action Parameter #2 – Gain	The position value is multiplied by the gain value. For example, if the gain is set to 10 and the position value is 1 mm (or any other unit), then the output voltage is 10 V.
Action Parameter #3 – Offset	The offset value is used to correct for any voltage that may already be present in the Analog output.

Analog output = Position value * gain + offset

Action parameter #4	This parameter is 0 by default.

DACSet.CurrentVelocity and **DACSet.SetpointVelocity:** This action sets a voltage on the Analog output relative to the actual (current) or theoretical (Setpoint) velocity. The gain and the offset are used to calibrate the output. This action makes most sense with events that have duration (Always, MotionState, ElementNumberState, etc.) as the analog output is updated at each servo cycle or at each profiler cycle as long as the event occurs. When used with events that have no duration (like MotionStart or MotionEnd), the analog output is only updated once and this value is kept until it is changed.

Action Parameter #1 – Positioner Name	This parameter defines the name of the positioner in which the Velocity value is used.
Action Parameter #2 – Gain	The Velocity value is multiplied by the gain value. For example if the gain is set to 10 and the velocity value is 1 mm/s (or any other velocity unit), then the output voltage is 10 V.
Action Parameter #3 – Offset	The offset value is used to correct for any voltage that may initially be present in the Analog output.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Analog output = Velocity value * gain + offset

Action parameter #4 This parameter must be 0 by default.

DACSet.SetpointAcceleration: 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 duration (Always, MotionState, ElementNumberState, etc.) as the analog output will be updated at each servo cycle or at each profiler cycle as long as the event lasts. When used with events that have no duration (like MotionStart or MotionEnd), the analog output is only updated once and keep this value until it is changed.

Action Parameter #1 – Positioner Name This parameter defines the name of the positioner in which the SetpointAcceleration is used to output in the analog output.

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 1 mm/s^2 then the output voltage will be 10 V.

Action Parameter #3 – Offset The offset value is used to correct for any voltage that may initially be present in the Analog output.

Analog output = SetpointAcceleration value * gain + offset

Action parameter #4 This parameter is 0 by default.

NOTE

The gain can be any constant value used to scale the output voltage and the offset value can be any constant value used to correct for any offset voltage in the analog output.

ExecuteTCLScript: This action executes a TCL script on an event.

Action Parameter #1 – TCL File Name This parameter defines the file name of the TCL program.

Action Parameter #2 – TCL Task Name Since several TCL scripts can run simultaneously different or even the same, the TCL Task Name is used to track individual TCL programs. For example, the TCL Task Name stops a particular program without stopping all other TCL programs that are running simultaneously.

Action Parameter #3 – TCL Argument List The Argument list is used to run the TCL scripts with input parameters. For the argument parameter, any input can be used (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 specify a number of loops inside the TCL script. A zero (0) for this parameter means there are no input arguments.

Action parameter #4 This parameter is 0 by default.

KillTCLScript: This action stops a TCL script on an event.

Action parameter #1 – Task name This parameter defines which TCL script is stopped. Since several TCL scripts can run

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

simultaneously, different or even the same script, the TCL Task Name is used to track individual TCL programs.

Action parameter #2 to #4 — These parameters are 0 by default.

GatheringOneData: This action acquires one data as defined by the function GatheringConfigurationSet. Different from the GatheringRun (see next action), which generates a new gathering file, the GatheringOneData appends the data to the current gathering file stored in memory. In order to store the data in a new file, first launch the function GatheringReset, which deletes the current gathering file from memory.

Action parameter #1 to #4 — These parameters are 0 by default.

GatheringRun: This action starts an internal data gathering. It requires that an internal gathering was previously configured with the function GatheringConfigurationSet. The gathering must be launched by a punctual event and does not work with events that have duration.

Action Parameter #1 – NbPoints — This parameter defines the number of data acquisitions. NbPoints multiplied by the number of gathered data types must be smaller than 1,000,000. For instance, if 4 types of data are collected, NbPoints can not be larger than 250,000 (4*250,000 = 1,000,000).

Action Parameter #2 – Divisor — This parameter defines the frequency of data gathering in relation to the servo frequency of the system (10 kHz). This parameter must be an integer and greater than or equal to 1. For instance, if the parameter is set to 10, then data gathering will take place every 10^{th} servo cycle or at a rate of 1 kHz (10 kHz/10) or at every 1 msec.

Action Parameter #3 and #4 — These parameters are 0 by default.

GatheringRunAppend: This action continues a gathering previously stopped with the action GatheringStop, see next action.

Action parameter #1 to #4 — These parameters are 0 by default.

GatheringStop: This action halts a data gathering previously launched by the action GatheringStart. Use the action GatheringRunAppend to continue data gathering. Note that the action GatheringStop does not automatically store the gathered data from the buffer to the flash disk of the controller. To store data, use the function GatheringStopAndSave. For more details about data gathering, refer to chapter 12: “Data Gathering”.

Action parameter #1 to #4 — These parameters are 0 by default.

ExternalGatheringRun: This action starts an external data gathering. It requires that an external data gathering was previously configured with the function GatheringExternalConfigurationSet. The gathering must be launched by a punctual event and does not work with events that have duration.

Action Parameter #1 – NbPoints — This parameter defines the number of data acquisitions. NbPoints multiplied by the number of gathered data types must be smaller than 1,000,000. For instance, if 4 types of data are collected, NbPoints can not be larger than 250,000 (4*250,000 = 1,000,000).

Action Parameter #2 – Divisor — This parameter defines every Nth number of the trigger input signal at which the gathering

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

	will take place. This parameter must be an integer and greater than 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 and #4	These parameters are 0 by default.

For further details on data gathering, see chapter 12: “Data Gathering”.

MoveAbort: This action stops (abort) a motion on an event. It is similar to sending a MoveAbort() function on the event. After stopping, the group is in the READY state.

Action Parameter #1 to #4	These parameters are 0 by default.

11.3 Functions

The following functions are related to event triggers:

- **EventExtendedConfigurationTriggerSet ()**: This function configures one or several events. In the case of several events, the different events are separated by a comma (,) in the argument list. Before activating an event, one or several actions must be configured with the function EventExtendedConfigurationActionSet(). Only then, the event and the associated action(s) can be activated with the function EventExtendedStart().
- **EventExtendedConfigurationTriggerGet ()**: This function returns the event configuration defined by the last EventExtendedConfigurationTriggerSet() function.
- **EventExtendedConfigurationActionSet ()**: This function associates an action to the event defined by the last EventExtendedConfigurationTriggerSet() function.
- **EventExtendedConfigurationActionGet ()**: This function returns the action configuration defined by the last EventExtendedConfigurationActionSet() function.
- **EventExtendedStart ()**: This function launches (activates) the last configured event and the associated action(s) defined by the last EventExtendedConfigurationTriggerSet() and EventExtendedConfigurationActionSet() and returns an event identifier. When activated, the XPS controller checks for the event at each servo cycle (or at each profiler cycle for those events that are motion related) and triggers the action when the event occurs. Hence, the maximum latency between the event and the action is equal to the servo cycle time of 100 µs or equal to the profiler cycle time of 400 µs for motion related events. For events with duration, it also means that the same action is triggered at each servo cycle, meaning every 100 µs, or at each profiler cycle, which is every 400 µs as long as the event occurs.

  Event triggers (and their associated action) are automatically removed after the event configuration has happened at least once and is no longer true. The only exception is if the event configuration contains any of the permanent events “Always” or “Trigger”. In this case the event trigger will always stay active. With the function EventExtendedRemove(), any event trigger can get removed.
- **EventExtendedWait ()**: This function halts a process (essentially by blocking the socket) until the event defined by the last EventExtendedConfigurationTriggerSet() occurs.
- **EventExtendedRemove ()**: This function removes the event trigger associated with the defined event identifier.
- **EventExtendedGet ()**: This function returns the event configuration and the action configuration associated with the defined event identifier.
- **EventExtendedAllGet ()**: This function returns, for all active event triggers, the event identifier, the event configuration and the action configuration. The details of the different event triggers are separated by a comma (,).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

11.4 Examples

Below is a table that shows possible events that can be associated with possible actions. Some of these examples however, may have unwanted results. Since the XPS controller provides great flexibility to trigger almost any action at any event, the user must be aware of the possible unwanted effects.

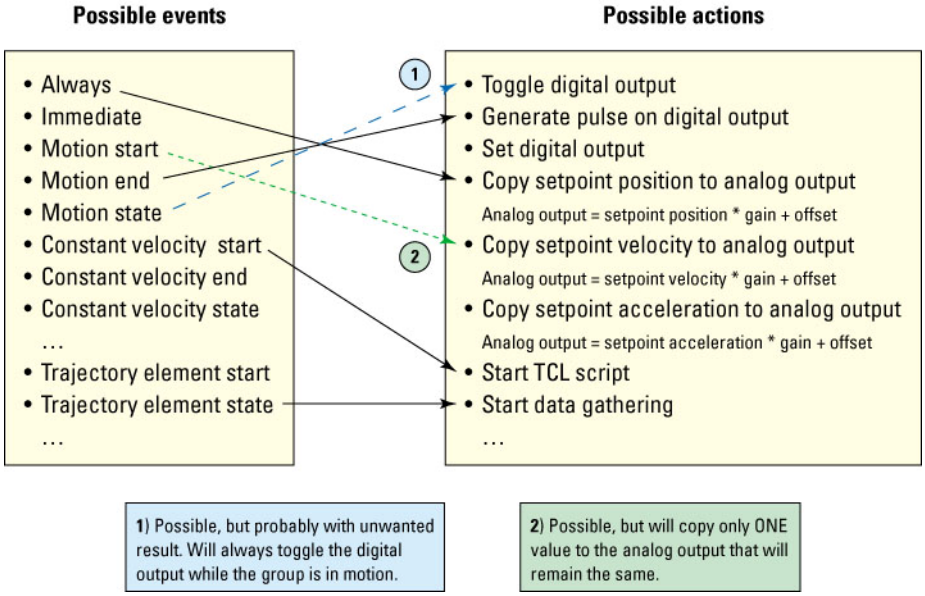


Figure 46: Possible Events.

Examples

1. **EventExtendedConfigurationTriggerSet (G1.P1.SGamma.ConstantVelocityStart, 0, 0, 0, 0)**

   EventExtendedConfigurationActionSet (GPIO1.DO.DOSet, 4, 4, 0, 0)

   EventExtendedStart()

   GroupMoveAbsolute (G1.P1, 50)

   In this example, when positioner G1.P1 reaches constant velocity, bit #3 on the digital output on connector number 1 is set to 1 (Note: 4 = 00000100). Note, that the state of the bit will not change when the constant velocity of the positioner has ended. In order to do so, a second event trigger would be required (see next example).

2. **EventExtendedConfigurationTriggerSet (G1.P1.SGamma.ConstantVelocityStart, 0, 0, 0, 0)**

   EventExtendedConfigurationActionSet (GPIO1.DO.DOSet, 4, 4, 0, 0)

   EventExtendedStart()

   EventExtendedConfigurationTriggerSet (G1.P1.SGamma.ConstantVelocityEnd, 0, 0, 0, 0)

   EventExtendedConfigurationActionSet (GPIO1.DO.DOSet, 4, 0, 0, 0)

   EventExtendedStart()

   GroupMoveAbsolute (G1.P1, 50)

   In this example, when positioner G1.P1 reaches constant velocity, bit #3 on the digital output on connector number 1 is set to 1 (Note: 4 = 00000100) and when the constant velocity of the positioner G1.P1 is over, bit #3 will be set to zero. Note, that the same effect can not be reached with the event name ConstantVelocityState.
   After both events have happened, the event triggers will get automatically removed. In order to trigger the same action at each motion, it is required to link the events

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

with the event “Always” (see next example). This link will avoid that the event trigger gets removed after it is not happening anymore.

3. **EventExtendedConfigurationTriggerSet (Always, 0, 0, 0, 0, G1.P1.SGamma.ConstantVelocityStart, 0, 0, 0, 0)**

   EventExtendedConfigurationActionSet (GPIO1.DO.DOSet, 4, 4, 0, 0)

   EventExtendedStart()

   EventExtendedConfigurationTriggerSet (Always, 0, 0, 0, 0, G1.P1.SGamma.ConstantVelocityEnd, 0, 0, 0, 0)

   EventExtendedConfigurationActionSet (GPIO1.DO.DOSet, 4, 0, 0, 0)

   EventExtendedStart()

   GroupMoveAbsolute (G1.P1, 50)

   GroupMoveAbsolute (G1.P1, -50)

   In this example, when positioner G1.P1 reaches constant velocity, bit #3 on the digital output on connector number 1 is set to 1 (Note: 4 = 00000100) and when the constant velocity of the positioner G1.P1 is over, bit #3 will be set to zero. Different from the previous example, adding the event “Always” avoids the event trigger being removed after the event is over. Hence, the state of the bit #3 will change with every beginning and with every end of the constant velocity state of a motion.

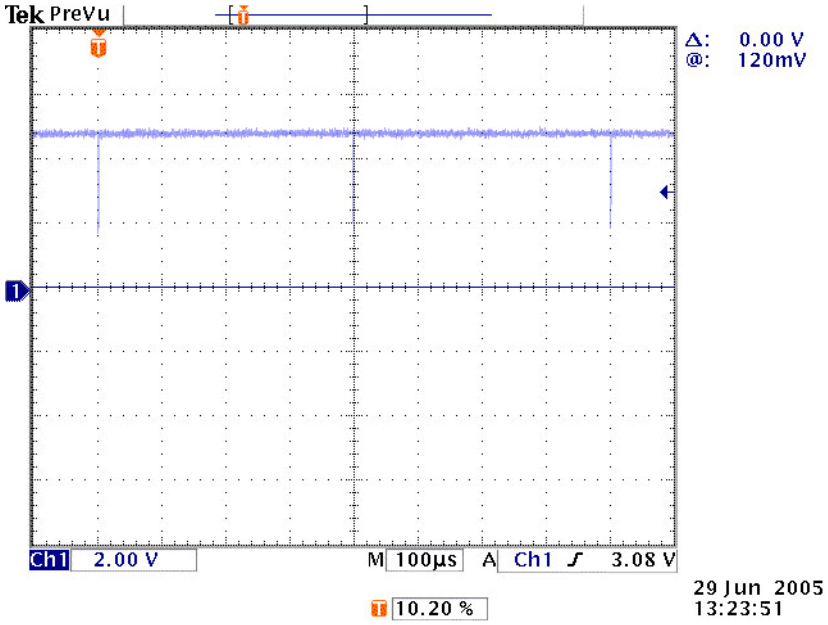
4. **EventExtendedConfigurationTriggerSet (G1.P1.SGamma.ConstantVelocityState, 0, 0, 0, 0)**

   EventExtendedConfigurationActionSet (GPIO1.DO.DOSet, 255, 0, 0, 0)

   EventExtendedStart()

   GroupMoveAbsolute (G1.P1, 50)

   In this example, during the constant velocity state of the positioner G1.P1, 1 µs pulses are generated on all 8 bits in the digital output on connector number 1, at every cycle of the motion profiler (Note: 255 = 11111111). The cycle time of the motion profiler is 400 µs, so pulses are generated every 400 µs (see picture below).



5. **EventExtendedConfigurationTriggerSet (Always, 0, 0, 0, 0)**

   EventExtendedConfigurationActionSet (GPIO2.DAC1.DACSet.SetpointPosition, 0.1, -10, 0, 0

   GPIO2.DAC2.DACSet.SetpointVelocity, 0.5, 0, 0, 0)

   EventExtendedStart()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In this example, the analog output #1 on GPIO2 will always output a voltage in relation to the SetpointPosition of the positioner G1.P1, and the output #2 on GPIO2 will always output a voltage in relation to the SetpointVelocity of the same positioner. The gain on output #1 is set to 0.1 V/unit and the offset to -10 V. This means that when the stage is at the position 0 units, a voltage of -10 V will be sent. When the stage is at the position 10 units, a voltage of -9V will be sent.
Here, the event "Always" means that these values will be updated every servo cycle, means every 0.1 ms. If instead of the event "Always", the event "Immediate" will be used, only the most recent values will be sent and kept. If instead of the event "Always", a motion related event such as MotionState is used, the update will only happen at every profiler cycle, or every 0.4 ms.

6. **TimerSet(Timer1,10000)**

   EventExtendedConfigurationTriggerSet (Timer1.Timer, 0, 0, 0, 0)

   EventExtendedConfigurationActionSet (GPIO1.DO.DOToggle, 255, 0, 0, 0)

   EventExtendedStart()

   EventExtendedRemove(1)

   The function Timer() sets the Timer1 at every 10,000th servo cycle, or at one second. Hence, in this example, every second all bits in the digital output on connector number 1 will be toggled (Note: 255 = 11111111). The event Timer is permanent. In order to remove the event trigger, use the function EventExtendedRemove() with the associated event identifier (1 in this case).

7. **MultipleAxesPVTPulseOutputSet(G1,2,20,1)**

   GatheringConfigurationSet(G1.P1.CurrentPosition)

   EventExtendedConfigurationTriggerSet(Always, 0,0,0,0,G1.PVT.TrajectoryPulse,0,0,0,0)

   EventExtendedConfigurationActionSet(GatheringOneData,0,0,0,0)

   EventExtendedStart()

   MultipleAxesPVTExecution(G1,Traj.trj,1)

   In this example, the generation of an output pulse every one second between the 2nd and the 20th element in the next PVT trajectory executed on the group G1 is first defined (function MultipleAxisPVTPulseOutputSet). Then, data gathering is defined (CurrentPosition of positioner G1.P1).

   Hence, in this example, with every trajectory pulse, one data point is gathered and appended to the current gathering file in memory. Here, adding the event TrajectoryPulse with the permanent event Always makes sure that the event trigger is always active. Without the event Always, only one data point will be gathered.

   This is because any event is automatically removed once it happens and does not happening in the next servo or profiler cycle (which is the case here as a pulse is only generated every one second).

   Please note that the action GatheringOneData appends data to the current data file. In order to store the data in a new file it is required to first launch the function GatheringReset() which deletes the current data file from memory.

8. **GatheringConfigurationSet(G1.P1.CurrentPosition)**

   EventExtendedConfigurationTriggerSet (G1.P1.SGamma.MotionStart,0,0,0,0)

   EventExtendedConfigurationActionSet(GatheringRun,20,1000,0,0)

   EventExtendedStart()

   GroupMoveAbsolute (G1.P1, 50)

   GatheringStopAndSave()

   In this example, an internal data gathering of 20 data points every 0.1 second (every 1000th servo cycle) is launched with the start of the next motion of the positioner

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

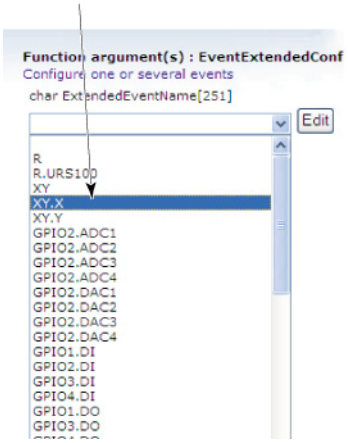
G1.P1. The type of data that gathered is defined with the function GatheringConfigurationSet (CurrentPosition of positioner G1.P1). To store the data from internal memory to the flash disk in the XPS controller, send the function GatheringStopAndSave(). The GatheringRun deletes the current data file in internal memory (in contrast to the GatheringOneData which appends data to the current file). Also, the function GatheringStopAndSave() stores the data file under a default name Gathering.dat on the flash disk of the XPS controller and will overwrite any older file of the same name in the same folder. Hence, make sure to store valuable data files under a different name before a GatheringStopAndSave().

NOTE

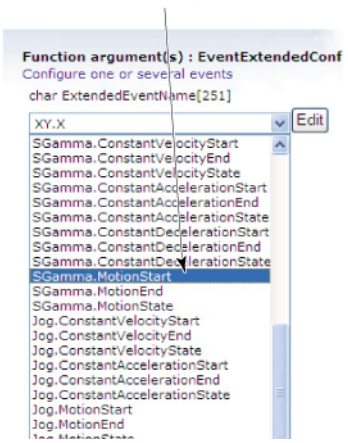
When using the function EventExtendedConfigurationTriggerSet() or EventExtendedConfigurationActionSet () from the terminal screen of the XPS utility, the syntax for one parameter is not directly accessible. For instance, for the event XY.X.SGamma.MotionStart, first select XY.X from the choice list. Then, click on the choice field again and select SGammaMotionStart. See also screen shots below.

For specifying more than one data type, use the ADD button. Select the next parameter as described above.

Step 1:
Select the positioner name and click.



Step 2:
Click in the choice field again and select the parameter name.



Step 3:
Define event parameters.
To add another event, click ADD, else click OK.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

12.0 Data Gathering

The XPS controller provides four methods for data gathering:

1. Time-based (internal) data gathering. With this method one data set is gathered for every n^{th} servo cycle.
2. Event-based (internal) data gathering. With this method one data set is gathered at an event.
3. Function-based (internal) data gathering. With this method one data set is gathered by a function.
4. Trigger-based (external) data gathering. With this method one data set is gathered for every n^{th} external trigger input (see also chapter 13.0: "Output Triggers").

Method 1, 2, and 3, these are also referred to as internal or servo cycle synchronous data gathering. With the trigger-based data gathering, this is also referred to as an external data gathering, as the event that triggers the data gathering or the receipt of a trigger input, is asynchronous to the servo cycle.

The time-based, the event-based and the function-based data gathering store the data in a common file called gathering.dat. The trigger-based (external) data gathering stores the data in a different file, called ExternalGathering.dat. The type of data that can be gathered differs also between the internal and the external data gathering.

Before starting any data gathering, the type of data to be gathered needs to be defined using the functions GatheringConfigurationSet() (in case of an internal data gathering) or ExternalGatheringConfigurationSet() (in case of an external data gathering). Refer to the Programmer's Manual and the Gathering functions for a complete list of data types.

During data gathering, new data is appended to a buffer. With the functions GatheringCurrentNumberGet() and GatheringExternalCurrentNumberGet(), the current number of data sets in this buffer and the maximum possible number of data sets that fits into this buffer can be recalled. The maximum possible number of data sets equals 1,000,000 divided by the number of data types belonging to one data set.

The function GatheringDataGet(index) returns one set of data from the buffer. Here, the index 0 refers to the 1st data set, the index (n-1) to the n-th data set. When using this function from the Terminal screen of the XPS utility, the different data types belonging to one data line are separated by a semicolon (;).

To save the data from the buffer to the flash disk of the XPS controller, use the functions GatheringStopAndSave() and GatheringExternalStopAndSave(). These functions will store the gathering files in the ..\Admin\Public folder of the XPS controller under the name Gathering.dat (with function GatheringStopAndSave() for internal gathering) or GatheringExternal.dat (with function GatheringExternalStopAndSave() for external gathering).

CAUTION



The functions GatheringStopAndSave() and GatheringExternalStopAndSave() overwrite any older files with the same name in the ..\Admin\Public folder. After a data gathering, it is required to rename or better, to relocate valid data files using an ftp link to the XPS controller (see chapter 5: "FTP connection").

A gathering file can have a maximum of 1,000,000 data entries and a maximum of 25 different data types. The first line of the data file contains the sample period in seconds (minimum period = 0.0001 s), the second line contains the names of the data type(s) and the other lines contain the acquired data. A sample file is shown below.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Gathering.dat

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

12.1 Time-Based (Internal) Data Gathering

The data for time-based gathering are latched by an internal interrupt related to the servo cycle of the XPS (10 kHz). The function GatheringConfigurationSet() defines the type of data that will be stored in the data file. The following is a list of all the data type(s) that can be collected:

PositionerName.CurrentPosition

PositionerName.SetpointPosition

PositionerName.FollowingError

PositionerName.CurrentVelocity

PositionerName.SetpointVelocity

PositionerName.CurrentAcceleration

PositionerName.SetpointAcceleration

PositionerName.CorrectorOutput

GPIO (ADC, DAC, DI, DO) See the Programmer's Guide for a list of all the GPIO Names of the Analog and Digital I/O.

The Setpoint values refer to the theoretical values from the profiler whereas the current values refer to the actual or real values of position, velocity and acceleration.

To gather information from the secondary positioner of a gantry, append ".SecondaryPositioner" to the positioner name. Example:

PositionerName.SecondaryPositioner.FollowingError

For details about gantry configurations, see chapter 4.9.

It is possible to start the gathering either by a function call or at an event. The following sequence of functions is used for a time-based data gathering started by a function call:

GatheringConfigurationSet()

GatheringRun()

The following sequence of functions is used to start a time-based data gathering at an event:

GatheringConfigurationSet()

EventExtendedConfigurationTriggerSet()

EventExtendedConfigurationActionSet()

EventExtendedStart()

A function triggers the action, for instance, a GroupMoveRelative().

When all data is gathered, use the function **GatheringStopAndSave()** to save the data from the buffer to the flash disk of the XPS controller.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Other functions associated with internal Gathering are:

GatheringConfigurationGet()

GatheringCurrentNumberGet()

GatheringDataGet()

GatheringDataMultipleLinesGet()

GatheringStop()

GatheringRunAppend()

See the Programmer's Manual for details about these functions.

NOTE

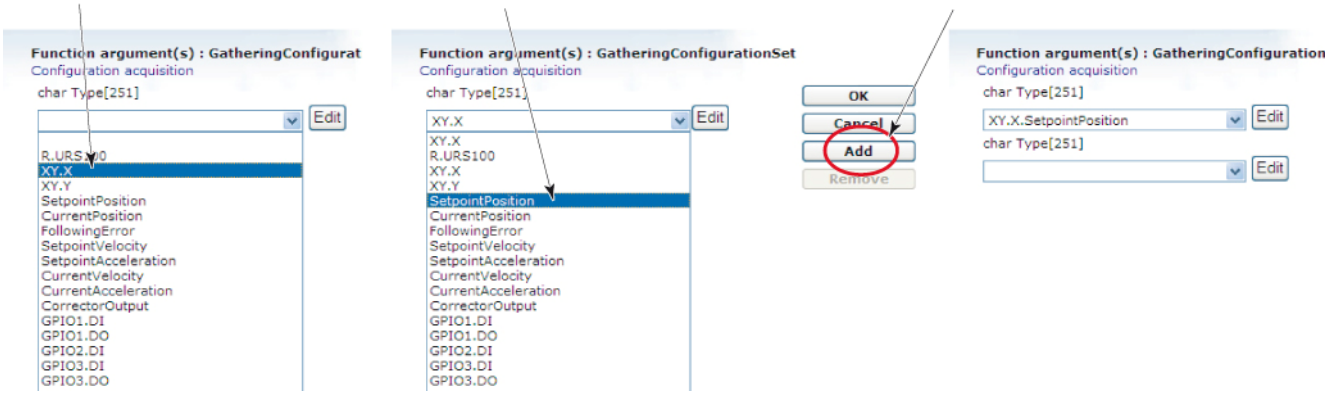
When using the function GatheringConfigurationSet() from the terminal screen of the XPS utility, the syntax for one parameter is not directly accessible. For instance, for the parameter XY.X.SetpointPosition, first select XY.X from the choice list. Then, click on the choice field again and select SetpointPosition. See also screen shots on the next page.

For specifying more than one data type, use the ADD button. Select the next parameter as described above.

Step 1:
Select the positioner name and click.

Step 2:
Click in the choice field again.
Select parameter name and click.

Step 3:
To add another parameter, press ADD.
Repeat steps 1 and 2.



Example 1

Using the terminal screen of the XPS utility, this example shows the sequence of functions to accomplish a time-based data gathering triggered at an event.

GroupInitialize(XY)

GroupHomeSearch(XY)

GatheringConfigurationSet(XY.X.SetpointPosition, XY.X.CurrentVelocity, XY.X.SetpointAcceleration)

The 3 data XY.X.SetpointPosition, XY.X.CurrentVelocity and XY.X.SetpointAcceleration will be gathered.

EventExtendedConfigurationTriggerSet (XY.X.SGamma.MotionStart,0,0,0,0)

EventExtendedConfigurationActionSet(GatheringRun,5000,10,0,0)

EventExtendedStart()

GroupMoveRelative(XY.X, 50)

GatheringStopAndSave()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In this example, gathering is started when the positioner XY.X starts its next motion using the Sgamma profiler, in this case with GroupMoveRelative() or possibly with GroupMoveAbsolute(). The types of data being collected are the Setpoint Position, Current Velocity and Setpoint Acceleration for the positioner XY.X. A total of 5000 data sets is collected, one data point every 10th servo cycles, or one data point every 10/10000 s = 0.001 s.

Example 2

Using the terminal screen of the XPS utility, this example shows the sequence of functions to accomplish a time-based data gathering started by a function call.

GroupInitialize(X)

GroupHomeSearch(X)

GatheringConfigurationSet(X.X.SetpointPosition, X.X.FollowingError)

GatheringRun (5000,10)

GroupMoveRelative (X, 10)

GatheringStop ()

GatheringStopAndSave ()

In this example, gathering is started by a function call. The SetpointPosition and FollowingError of the positioner XY.X are gathered at a rate of 1 kHz (every 10th servo cycle, 10 kHz servo cycle rate). Data gathering is stopped after the relative move is completed.

Gathering will stop automatically once the number of points specified has been collected. However, data will not be saved automatically to a file. The function **GatheringStopAndSave()** must be used to save the data to a file.

It is also possible to halt data gathering at an event. To do so, define another event trigger and assign the action GatheringStop to that event. Use another event trigger and assign the action GatheringRunAppend to continue with gathering. For details, see chapter 11.0: “Event Triggers“.

Note

The function GatheringRun() always starts a new internal data gathering and deletes any previous internal gathering data hold in the buffer. If you want to append data to the file, use the function GatheringRunAppend() instead.

12.2 Event-Based (Internal) Data Gathering

The event-based gathering provides a method to gather data at an event. For instance, gathering data at a certain value of a digital or analog input, during a constant velocity state of a motion or on a trajectory pulse.

The event-based data gathering uses the same file as the time-based and the function based data gathering (see sections 12.1 and 12.3). However, unlike the time-based gathering, the event-based gathering appends data to the existing file in memory. This allows gathering of data during several periods or even with different methods in one common file, see examples. To start data gathering in a new file, use the function GatheringReset(), which deletes the current gathering file from memory.

The data type(s) that can be collected with event-based gathering are the same as data for time-based and function-based gathering:

PositionerName.CurrentPosition

PositionerName.SetpointPosition

PositionerName.FollowingError

PositionerName.CurrentVelocity

PositionerName.SetpointVelocity

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PositonerName.CurrentAcceleration

PositionerName.SetpointAcceleration

PositionerName.CorrectorOutput

GPIO (ADC, DAC, DI, DO) See Programmer's manual for a list of all the GPIO Names for the Analog and Digital I/O.

The Setpoint values refer to the theoretical values from the profiler where as the current values refer to the actual or real values of position, velocity and acceleration.

To gather information from the secondary positioner of a gantry, append ".SecondaryPositioner" to the positioner name. Example:

PositionerName.SecondaryPositioner.FollowingError

For details about gantry configurations, see chapter 4.9.

The following sequence of functions is used in event-based data gathering:

GatheringReset()

GatheringConfigurationSet()

EventExtendedConfigurationTriggerSet()

EventExtendedConfigurationActionSet(GatheringOneData,0,0,0,0)

EventExtendedStart()

…

Use the function GatheringStopAndSave() to store the gathered file from the buffer to the flash disk of the XPS controller.

Other functions associated with the event-based gathering are:

GatheringConfigurationGet()

GatheringCurrentNumberGet()

GatheringDataGet()

Please refer to the programmer's manual for details.

Example 1

GatheringReset()

Deletes gathering buffer in memory.

GatheringConfigurationSet(XY.X.CurrentPosition, XY.Y.CurrentPosition, GPIO2.ADC1)

The 3 data XY.X.CurrentPosition, XY.Y.CurrentPosition and GPIO2.ADC1 will be gathered.

EventExtendedConfigurationTriggerSet(GPIO2.ADC1.ADCHighLimit, 5,0,0,0)

EventExtendedConfigurationActionSet(GatheringOneData,0,0,0,0)

EventExtendedStart()

Data gathering starts when the value of the GPIO2.ADC1 exceeds 5 Volts. One set of data will be gathered at each servo cycle or every 100 µs (as the event is checked at each servo cycle). Data gathering automatically stops when the value of the GPIO2.ADC1 falls below 5 V again and the event is automatically removed (see chapter 11.0: "Event Triggers" for details).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example 2

TimerSet(Timer1, 10)

Sets the timer 1 to 10 servo ticks, means every 1 ms.

GatheringReset()

Deletes gathering buffer from memory.

GatheringConfigurationSet(XY.X.CurrentPosition, XY.Y.CurrentPosition, GPIO2.ADC1)

The 3 data XY.X.CurrentPosition, XY.Y.CurrentPosition and GPIO2.ADC1 will be gathered.

EventExtendedConfigurationTriggerSet(Timer1,0,0,0,0, GPIO2.ADC1.ADCHighLimit,5,0,0,0)

EventExtendedConfigurationActionSet(GatheringOneData,0,0,0,0)

EventExtendedStart()

Different from the previous example, here the event ADCHighLimit is linked to the event Timer1. This has two effects. First, the event becomes permanent as the event timer is permanent. Second, one set of data is gathered only every 10 ms (combination of events must be true). For details on the event definition, please see chapter 11.0: “Event Triggers“.

As a result, one set of data is gathered every 10 ms whenever the value of the GPIO2.ADC1 exceeds 5 Volts.

Example 3

TimerSet(Timer1, 10)

Sets the timer 1 to 10 servo ticks, means every 1 ms.

GatheringReset()

Deletes gathering buffer from memory.

GatheringConfigurationSet(XYZ.X.CurrentPosition, XYZ.Y.CurrentPosition, XYZ.Z.CurrentPosition)

EventExtendedConfigurationTriggerSet(Timer1,0,0,0,0, XYZ.Spline.TrajectoryState,0,0,0,0)

EventExtendedConfigurationActionSet(GatheringOneData,0,0,0,0)

EventExtendedStart()

In this example, during the execution of the next spline trajectory on the group XYZ, one set of data will be gathered every 10 ms. In contrast to time-based gathering, which allows programming of a similar function, data gathering will automatically stop at the end of the trajectory. Also, it is not needed to define the total number of data sets that will be gathered.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

12.3 Function-Based (Internal) Data Gathering

Function-based gathering provides a method to gather one set of data using a function. It uses the same data file as the time-based and the event-based data gathering, see chapters 13.1 and **Erreur ! Source du renvoi introuvable.** for details. At receipt of the function, one set of data is appended to the gathering file in memory.

The data type(s) that can be collected with the event-based gathering are the same as for the time based and the event-based gathering, see chapter 12.1 and 12.2 for details.

Example

GatheringReset()

Deletes gathering buffer.

GatheringConfigurationSet(XY.X.CurrentPosition, XY.Y.CurrentPosition)

The 2 data XY.X.CurrentPosition and XY.Y.CurrentPosition will be gathered.

GatheringDataAcquire()

Gathers one set of data.

GatheringCurrentNumberGet()

This function will return 1, 500000; 1 set of data acquired, max. 500 000 sets of data can be acquired.

GatheringDataAcquire()

GatheringDataAcquire()

GatheringCurrentNumberGet()

This function will return 3, 500000; 3 sets of data acquired, max. 500,000 sets of data can be acquired.

12.4 Trigger-Based (External) Data Gathering

The trigger-based data gathering allows acquiring position and analog input data at receipt of an external trigger input (TRIG IN connector at the XPS, see section 26.0 for more details).

The position data is latched by dedicated hardware. The jitter between the trigger signal and the acquisition of the position data is less than 50 ns. The analog inputs, however, are only latched by an internal interrupt at a rate of 10 kHz and the XPS will store the most recent value. Hence, the acquired analog input data might be up to 100 µs old.

NOTE

There must be a minimum time of 100 µs between two successive trigger inputs.

The data of the trigger-based (external) data gathering is stored in a file named ExternalGathering.dat, which is different from the file used for the internal data gathering (Gathering.dat). Hence, internal and external data gathering can be used at the same time.

The function GatheringExternalConfigurationSet() defines which type of data will be gathered and stored in the data file. The following data types can be collected:

PositionerName.ExternalLatchPosition and
PositionerName.SecondaryPositioner.ExternalLatchPosition
(for secondary positioners of gantries, see chapter 4.9 for details).

These positions refer to the uncorrected encoder position, meaning no error corrections are taken into account. For devices with RS422 differential encoders, the resolution of the position information is equal to the encoder resolution.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For devices with sine/cosine 1Vpp analog encoder interface, the resolution is equal to the encoder scale pitch divided by the value of the positioner hard interpolator, see function PositionerHardInterpolatorFactorGet(). Its value is set to 20 by default; the maximum allowed value is 200. Please refer to the Programmer's Manual for details.

The external latch positions require that the device has an encoder. No position data can be latched with this method for devices that have no encoder.

GPIO2.ADC1 to GPIO.ADC4
(referring to the 4 analog input channels on the GPIO2)

The following sequence of functions is used for a trigger-based data gathering:

GatheringExternalConfigurationSet()

EventExtendedConfigurationTriggerSet()

EventExtendedConfigurationActionSet()

EventExtendedStart()

Other functions associated with event-based gathering are:

GatheringConfigurationGet()

GatheringCurrentNumberGet()

GatheringExternalDataGet()

Please refer to the Programmer's Manual for details.

Example

GatheringExternalConfigurationSet(XY.X.ExternalLatchPosition, GPIO2.ADC1)

EventExtendedConfigurationTriggerSet(Immediate,0,0,0,0)

EventExtendedConfigurationActionSet(ExternalGatheringRun,100,2,0,0)

EventExtendedStart()

In this example, a trigger-based (external) gathering is started immediately (with the function EventExtendedStart()). The types of data being collected are the XY.X encoder position and the value of the GPIO2.ADC1. A total of 100 data sets are collected; one set of data at each second trigger input. Gathering will stop automatically after the 100th data acquisition. Use the function **GatheringExternalStopAndSave()** to save the data to a file. The file format is the same as for internal data gathering.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

13.0 Output Triggers

External data acquisition tools, lasers, and other devices can be synchronized to the motion. For this purpose, the XPS features one dedicated trigger output per axis, see Appendix E, PCO connector for details. The XPS can be configured to either output distance spaced pulses, AquadB encoder signals, or time spaced pulses on this connector.

In the distance spaced configuration, one output pulse is generated when crossing a defined position and a new pulse is generated at every defined distance until a maximum position has been reached. In most cases, this mode provides the most precise synchronization of the motion to an external tool.

In the AquadB configuration, AquadB encoder signals are output on the PCO connector. These signals can be provided either always or only if the positioner is within a defined position window. When used with stages that feature a digital encoder (AquadB) as opposed to a SinCos encoder (AnalogInterpolated), the AquadB configuration essentially provides an image of the encoder signals on the PCO connector.

In the time flasher configuration, an output pulse is generated when crossing a defined position and a new pulse is generated at a defined time interval until a maximum position has been reached. In some cases, this mode can provide an even more precise synchronization of the motion to an external tool, in particular if the variation of the speed multiplied with the time interval is smaller than the error of the encoder signals during the same period.

Dedicated hardware is used to check the position crossing and the time interval to attain less than 50 ns latency between the position crossing and the trigger output.

For the distance spaced pulses configuration, time flasher configuration or AquadB signals on PCO connector configuration, it is recommended to calibrate the position compare before all PCO pulses generation. It is also recommended to set the position compare hardware to the scanning range you intend to use to get the best performances (*refer to Section 13.3 for details*).

In addition and independent from the above, the XPS controller can output distance spaced pulses on Line-arc trajectories and time spaced pulses on PVT trajectories. In these cases, the distances/time intervals are checked on the servo cycle and a resolution of 100 µs is provided.

13.1 Triggers on Line-Arc Trajectories

This capability outputs pulses at constant trajectory length intervals on Line-Arc-Trajectories. The pulses are generated between a start length and an end length. All lengths are calculated in an orthogonal XY plane. The StartLength, EndLength, and PathLengthInterval refer to the Setpoint positions.

The trajectory length is calculated at a rate of 10 kHz. This means that the resolution of the trajectory length is 0.0001 * trajectory velocity. For a trajectory velocity of 100 mm/s for instance, the resolution of the trajectory length is 10 µm. If the programmed PathLengthInterval is not a multiple of this resolution, the pulses can be off from the ideal positions by a maximum ± half of this resolution.

Two signals are provided:

GPIO2, pin11, Window: A constant 5 V signal is sent between the StartLength and the EndLength.

GPIO2, pin12, Pulse: A 1 µs pulse with 5 V peak voltage is sent every PathLengthInterval.

For details about the XPS I/O connectors, see appendix, section 22.2.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To define the StartLength, EndLength, and PathLengthInterval, use the function **XYLineArcPulseOutputSet()**.

Example

XYLineArcPulseOutputSet(XY, 10, 30, 0.01)

One pulse will be generated every 10 µm on the next Line-Arc Trajectory between 10 mm and 30 mm.

XYLineArcVerification(XY, Traj.trj)

Loads and verifies the trajectory Traj.trj

XYLineArcExecution(XY, Traj.trj, 10, 100, 1)

Executes the trajectory at a trajectory speed of 10 mm/s and with a trajectory acceleration of 100 mm/s one time.

Please note, that the pulse output settings are automatically removed when the trajectory is over. Hence, with the execution of every new trajectory, it is also required to define the pulse output settings again.

It is also possible to use the trajectory pulses and the pulse window state as events in the event triggers (see section 11.0: "Event Triggers" for details). This allows the gathering of data on a trajectory at constant length intervals.

Example

XYLineArcPulseOutputSet(XY, 10, 30, 0.01)

One pulse every 10 µm will be generated on the Line-Arc Trajectory between 10 mm and 30 mm.

XYLineArcVerification(XY, Traj.trj)

Loads and verifies the trajectory Traj.trj

GatheringConfigurationSet(XY.X.CurrentPosition, XY.Y.CurrentPosition, GPIO2.ADC1)

Configures data gathering to capture the current positions of the XY.X and the XY.Y and the analog input GPIO2.ADC1

EventExtendedConfigurationTriggerSet(Always, 0,0,0,0,XY.LineArc.TrajectoryPulse,0,0,0,0)

Triggers an action for every trajectory pulse. The link of the event TrajectorPulse with the event Always is important to make the event permanent. Otherwise, the event will be removed after the first pulse.

EventExtendedConfigurationActionSet(GatheringOneData,0,0,0,0)

Defines the action; gathers one set of data each trajectory pulse.

EventExtendedStart()

Starts the event trigger.

XYLineArcExecution(XY, Traj.trj, 10, 100, 1)

Executes the trajectory at a trajectory speed of 10 mm/s and a trajectory acceleration of 100 mm/s one time.

GatheringStopAndSave()

Saves the gathering data from memory into a file gathering.dat in the ..admin/public folder of the XPS.

In this example, one set of data will be gathered on the trajectory between length 10 mm and 30 mm at constant trajectory length intervals of 10 µm.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

13.2 Triggers on PVT Trajectories

This capability outputs pulses at constant time intervals on a PVT trajectory. The pulses are generated between a first and a last trajectory element (see 9.3, PVT Trajectories for details). The minimum possible time interval is 100 µs.

Two signals are provided:

GPIO2, pin11, Window: A constant 5 V signal is sent between the beginning of the first and the end of the last trajectory element.

GPIO2, pin12, Pulse: A 1 µs pulse with 5V peak voltage is sent for every time interval

For details about the XPS I/O connectors, see appendix, section 22.2.

To define the first element, the last element and the time interval, use the function **MultipleAxesGroupPVTPulseOutputSet()**.

Example 1

MultipleAxesGroupPVTPulseOutputSet (Group1, 3, 5, 0.01)

One pulse will be generated every 10 ms between the start of the 3rd element and the end of the 5th element.

MultipleAxesPVTVerification(Group1, Traj.trj)

Loads and verifies the trajectory Traj.trj

MultipleAxesPVTExecution(XY, Traj.trj, 1)

Executes the trajectory Traj.trj one time.

Note that the pulse output settings are automatically removed when the trajectory is over. Hence, with the execution of every new trajectory, the pulse output settings must be defined again.

It is also possible to use the trajectory pulses and the pulse window state as events in the event triggers (see section 11.0: “Event Triggers“ for details). This allows the gathering of data on a trajectory.

Example 2

MultipleAxesPVTPulseOutputSet(Group1, 3, 5, 0.01)

One pulse will be generated every 10 ms between the start of the 3rd element and the end of the 5th element.

MultipleAxesPVTVerification(Group1, Traj.trj)

Loads and verifies the trajectory Traj.trj

GatheringConfigurationSet(Group1.P.CurrentPosition, GPIO2.ADC1)

Configures data gathering to capture the current position of the Group1.P positioner and the analog input GPIO2.ADC1

EventExtendedConfigurationTriggerSet(Always, 0,0,0,0,Group1.PVT.TrajectoryPulse,0,0,0,0)

Triggers an action for every trajectory pulse. The link of the event TrajectorPulse with the event Always is important to make the event permanent. Otherwise, the event will be removed after the first pulse.

EventExtendedConfigurationActionSet(GatheringOneData,0,0,0,0)

Defines the action; gathers one set of data each trajectory pulse.

EventExtendedStart()

Starts the event trigger

MultipleAxesPVTExecution(XY, Traj.trj, 1)

Executes the trajectory Traj.trj one time.

GatheringStopAndSave()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Saves the gathering data from memory in a file gathering.dat in the ..admin/public folder of the XPS.

In this example, one set of data will be gathered every 10 ms on the trajectory between the start of the 3rd and the end of the 5th element.

13.3 Distance, Time Spaced Pulses or AquadB Position Compare

13.3.1 Position compare settings and limits of use

13.3.1.1 Position compare calibration

EncoderIndexOffset (*stages.ini*) is a configuration parameter (measured in position unit like mm, deg. …), used to correct the offset between stage raw encoder position and stage hard interpolator position values.

To measure *EncoderIndexOffset* value, do the followings:
- Set *EncoderIndexOffset = 0* in *stages.ini* then reboot the controller.
- Send the following commands to the controller:
  GroupInitialize() then *GroupHomeSearch()*.
  PositionerHardInterpolatorPositionGet()
  Example: *PositionerHardInterpolatorPositionGet((XY.X, double *)*
  Controller response: 0,*Value1* (example 0,0.001).
- Set *EncoderIndexOffset = Value1* in *stages.ini* then reboot the controller once again.
  Send the same commands: *GroupInitialize()*, *GroupHomeSearch()* then *PositionerHardInterpolatorPositionGet()*.
  Controller response: 0,*Value2* (example 0, 2.71051e-20). *Value2* must be closed to 0, meaning that the position compare is now calibrated.

Note

It is not mandatory to calibrate the PCO and its accuracy is application dependent.

13.3.1.2 Valid settings as a function of scan velocity and PCO pulse settling time

- Determine *PCO encoder frequency*:
  AquadB encoder:
  PCO encoder frequency = ScanVelocity / EncoderResolution
  Analog Sin/Cos encoder:
  *PCO encoder frequency = ScanVelocity * HardInterpolatorFactor / EncoderScalePitch*

  Example: *ScanVelocity* = 10 mm/s, *EncoderScalePitch* = 0.004 mm, *HardInterpolatorFactor* = 200 => *PCO encoder frequency* = 10 * 200 / 0.004 = 500000 = 500 kHz

- The valid settings are shown in the following table:

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

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note

When changing the PCO pulse settling time you must limit the maximum velocity of the stage accordingly otherwise you will loose the PCO position and generate the wrong number of pulses at wrong positions. As per the above table, if you set the pulse settling time to 4 µs, the maximum *PCO encoder frequency* needs to be limited to less than 0.25 / 4x 10^{-6} = 62.5 kHz.

So, if *EncoderScalePitch* = 0.004 mm and *HardInterpolatorFactor* = 200 then the *ScanVelocity* must ≤ 62.5 x 10^{3} * 0.004 / 200 = 1.25 mm/s, otherwise the PCO will not work properly.

13.3.2 Even Distance Spaced Pulses Position Compare

In the even distance spaced pulse configuration, one first output pulse is generated when the positioner enters the defined position window. This is independent of the positioner entering the window from the minimum position or from the maximum position. From this first pulse position, a new pulse is generated at every position step until the stage exits the window.

NOTE

To make sure that the trigger pulses are always at the same positions independent of the positioner entering the window from the minimum or from the maximum window position, the difference between the minimum and the maximum window position should be an integer multiple of the position step.

The duration of the trigger pulse is 200 nsec by default and can be modified using the function **PositionerPositionComparePulseParametersSet (PositionerName, PCOPulseWidth, EncoderSettlingTime)**. Possible values for PCOPulseWidth are: 0.2 (default), 1, 2.5 and 10 (µs). Please note, that only the falling edge of the trigger pulse is precise and only this edge should be used for synchronization irrespactable from the PCOPulseWidth setting. Note also, that the duration of the pulse detected by your electronics may be longer depending on the time constant of your RC circuit. Successive trigger pulses should have a minimum time lag equivalent to the PCOPulseWidth time multiplied two.

The second parameter, EncoderSettlingTime applies a filter to the encoder signals for the trigger pulse generation. Possible values are: 0.075 (default), 1, 4, 12 (µs). The setting of this EncoderSettlingTime should be done in relation to the application, in particular speed and encoder resolution, and the encoder/position noise. For most applications, the default value works fine. At very low speed, with high encoder resolution, and significant encoder/position noise, however, it may be possible that additional trigger pulses are generated where no trigger pulse should be generated from the application. In these cases, a higher value setting for the EncoderSettlingTime could avoid these unwanted extra pulses. The value for the EncoderSettlingTime, however, should not exceed the value for the Encoder resolution divided by the speed. Please note also, that the EncoderSettlingTime adds a nominal delay between the encoder transition and the trigger pulse.

Example

With XM stages and the hardware interpolator set to 200 (see function PositionerHardInterpolatorFactorSet ()) the resolution of the trigger pulses is 20 nm (4 µm encoder scale pitch / 200). At continuous speed motion with 20 µm/s speed, the nominal time between successive encoder counts is 1 ms (20 nm / 20 µm/s). In a not optimum environment of the XM stages, it is possible, that the actual position detected by the trigger circuitry is not continuously increasing, but flickering around one encoder count (20 nm) from time to time. When using the default setting for the EncoderSettlingTime (0.075 µs) under these conditions, it is very likely that more than

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

one trigger pulse is generated (since the stage, seen by the controller, is moving back and forth). A higher value setting for the EncoderSettlingTime could avoid these unwanted and unpredictable extra trigger pulses in this case.

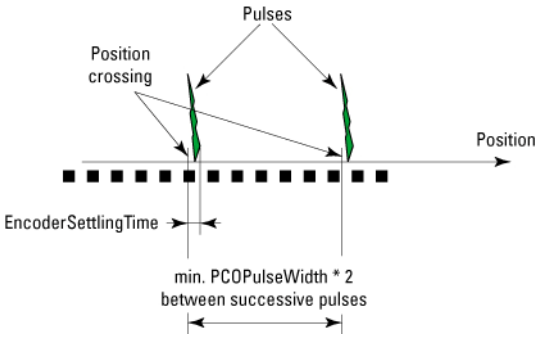


Figure 47: Position Compare Output.

The following functions are used to configure the distance spaced pulses:

PositionerPositionCompareSet

PositionerPositionCompareGet

PositionerPositionCompareEnable

PositionerPositionCompareDisable

The function PositonerPositonCompareSet() defines the position window and the distance for the trigger pulses. It has four input parameters:

Positioner Name

Minimum Position

Maximum Position

Position Step

To enable the distance spaced pulses, the function PositionerPositionCompareEnable() must be sent.

Example

GroupInitialize(MyStage)

GroupHomeSearch(MyStage)

PositionerPositionCompareSet(MyStage.X,5, 25, 0.002)

PositonerPositionCompareEnable(MyStage.X)

PositionerPositionCompareGet(MyStage, &MinimumPosition, &MaximumPosition, &PositionStep, &EnableState)

This function returns the parameters previously defined, the minimum position 5, the maximum position 25, the position step 0.002 and the enabled state (1=enabled, 0 =disabled).

GroupMoveAbsolute(MyStage,30)

PositionerPositionCompareDisable(MyStage.X)

The group has to be in a READY state for the position compare to be enabled. Also, the PositionerPositionCompareSet() function must be completed before the PositionerPositionCompareEnable() function. In this example, one trigger pulse is generated every 0.002 mm between the minimum position of 5 mm and the maximum position of 25 mm. The first trigger pulse will be at 5 mm and the last trigger pulse will be at 25 mm.

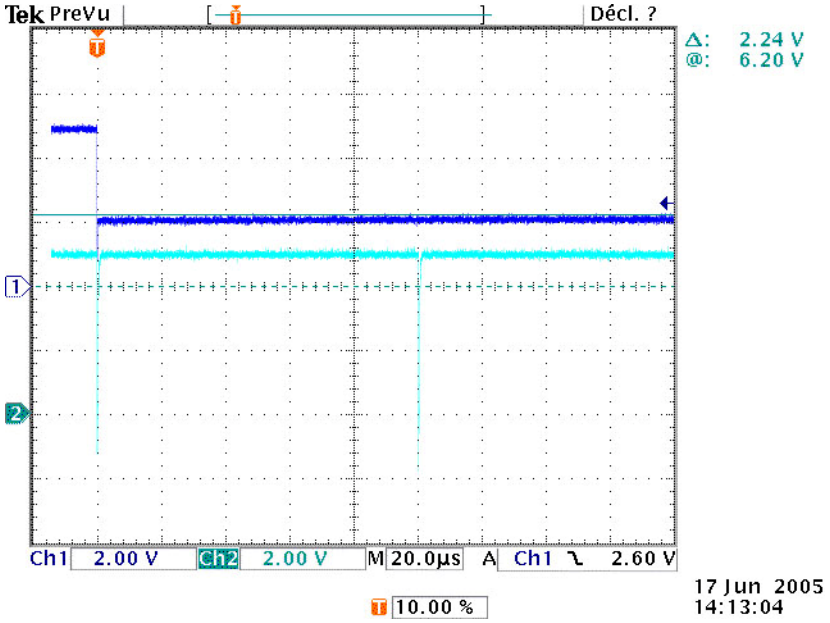
The output pulses are accessible from the PCO connector at the back of the XPS controller, See appendix E, PCO connector, for details.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

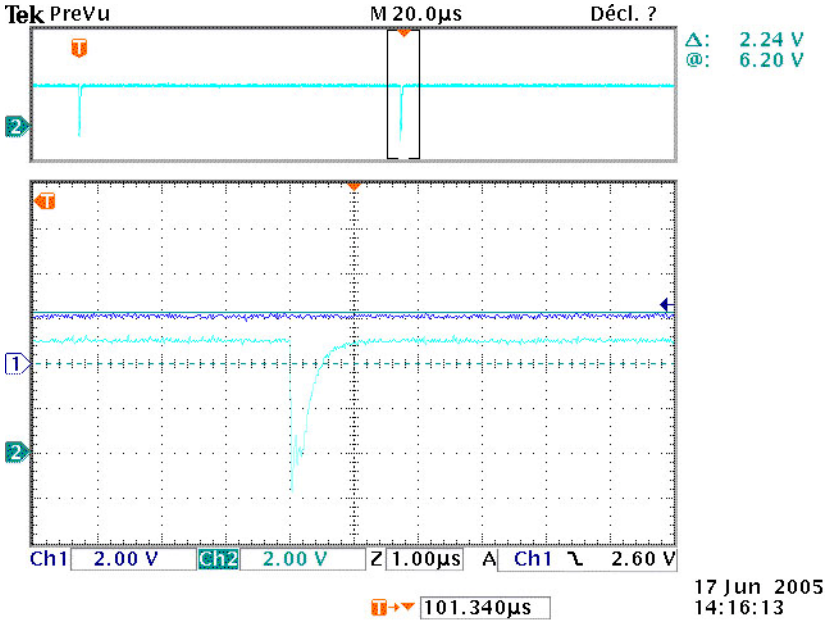
This table summarizes the results of the example above:

Position of the stage	Pulse enable 1 state	Pulse 1 activation	Explanation
0	0	No	Position compare not enabled
5	1	Yes	Position compare enabled, first pulse
5…25	1	Yes	One pulse every 0.002 mm
25	1	Yes	Last pulse
25.002	0	No	Position compare disabled
30	0	No	Position compare disabled

The figure below shows actual screen shots from an oscilloscope for the example above. The enable window is displayed in ch1 and the pulses in ch2:



At position 5 mm, the position compare output functionality becomes active and the first pulse is generated. Then, pulses are generated every 2 µm which equals a time span of 100 µs at a speed of 20 mm/s (2 µm/20 mm/s = 100 µs).



This second picture shows a zoom of the second pulse. The duration of the pulse should be 200 ns, however the pulse duration can be longer as the rising edge is related to the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

time constant of the RC circuit used. Please note that only the falling edge of the pulse is precise and should be used for synchronization purposes.

NOTE

The parameters PositionStep, MinimumPosition, and MaximumPosition (specified with the function PositionerPositionCompareSet) are rounded to the nearest detectable trigger position. When using the Position Compare function with AquadB encoders, the trigger resolution is equal to the EncoderResolution of the positioner specified in the stages.ini. When using the Position Compare function with AnalogInterpolated encoders, the trigger resolution is equal to the EncoderScalePitch defined in the stages.ini divided by the interpolation factor defined by the function PositionerHardInterpolatorFactorSet.

AnalogInterpolated encoder

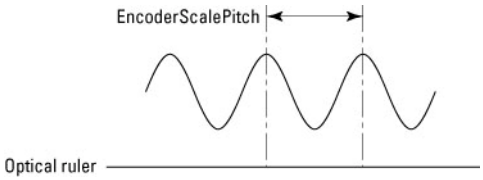


Figure 48: AnalogInterpolated Encoder.

$$\textbf{Trigger resolution} = \frac{\text{EncoderScalePitch}}{\text{PositionerHardInterpolatorFactor}}$$

Trigger pulses



Figure 49: Trigger Pulses.

MinimumPosition, MaximumPosition, and PositionStep should be multiples of the Trigger resolution. If not, rounding to the nearest multiple value is made.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

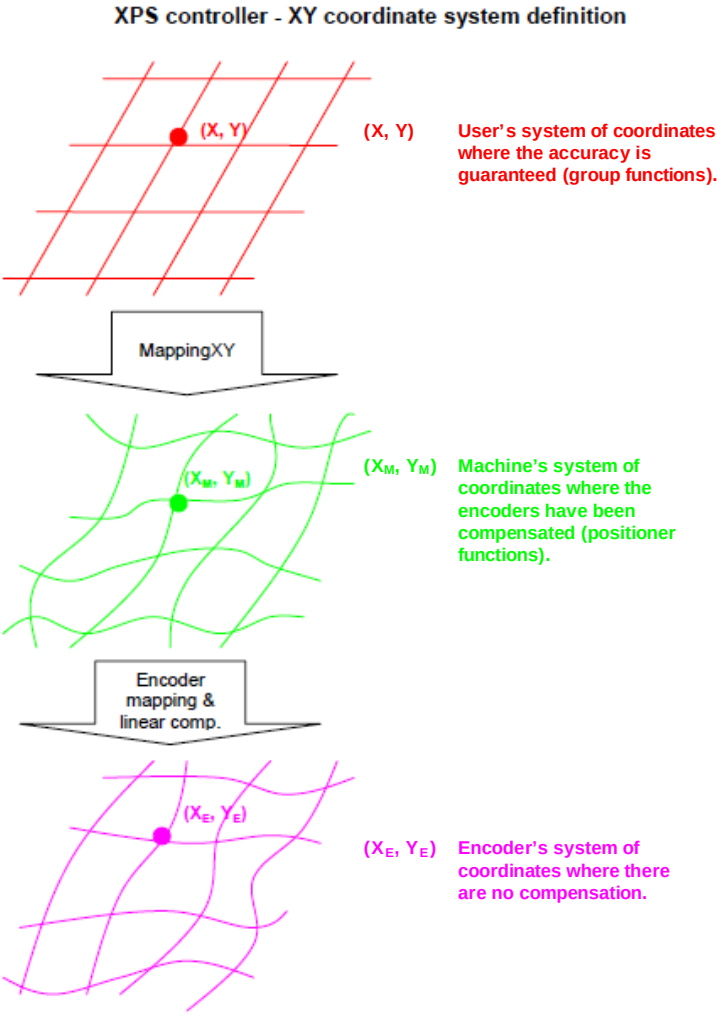
13.3.3 Compensated Position Compare

This feature requires hardware boards to be E4323x or later. Older hardware will return an error. This feature is used to output a pulse each time the stage moves over user predefined positions.

13.3.3.1 XPS system of coordinates

To explore the details of the XPS coordinate system, use the example of the XY group but the same is true for the other groups.

The firing positions are defined in the called user's system of coordinates (X, Y). The controller will convert the (X, Y) coordinates to raw encoder positions (X_E, Y_E) to take into account the group mapping, the encoder mapping and the encoder linear compensation to accurately fire the pulses at the requested positions.



To know the positions in the different systems of coordinates, the following functions are provided:

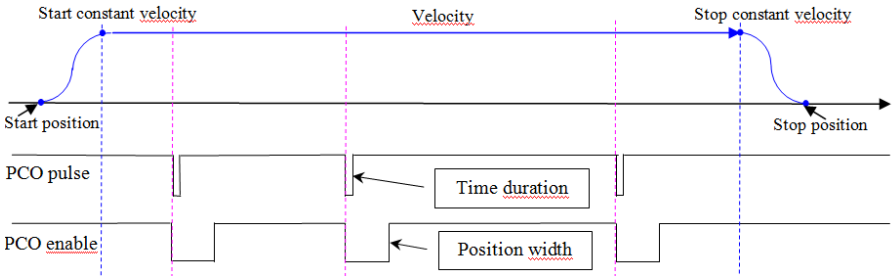
- *GroupPositionCorrectedProfilerGet*() function has as input a (X, Y) position in the user's system of coordinates and will output the (X_M, Y_M) position in the machine's system of coordinates by applying the XY mapping compensation.
- *GroupPositionPCORawEncoderGet*() function has as input a (X, Y) position in the user's coordinate system and will output the (X_E, Y_E) position in the encoder's system of coordinates without any compensation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

13.3.3.2 CIE08 compensated position compare signals definition

If the CIE08 compensated PCO pulses generation is activated, the PCO pulses will be generated at each predefined position with a pulse time duration that can be set with the *PositionerPositionComparePulseParametersSet()* function *(cf. XPS Programmer's Manual for details)*.

The PCO enable output signal will be generated at the same position as the PCO pulse and its width will be adjusted as a position value.



The upper limit for the pulse rate will be the distance travelled by the stage over a servo cycle with some margin to avoid any pulse overrun.

The minimum distance between trigger positions can be calculated as:

*MinimumTriggerPulseDistance > 1.2 * ScanningVelocity * ServoCycleTime*

For example for a stage running at 300 mm/s and a servo cycle of 100 µs, the minimum distance between trigger pulses must be bigger than 36 µm.

The margin to take in account will depend on many parameters such as the speed stability.

13.3.3.3 CIE08 compensated position compare scanning process description

13.3.3.3.1 Scan preparation

- Add the following lines in XPS controller /Admin/Config/system.ini, [ScanningPositioner] section:

  CIE08CompensatedPCOMode = Enabled ; Enabled or Disabled

  CIE08CompensatedPCOMaximumDataNumber = ... ; Value must <= 1000000

- Reboot the controller.
- Set hard interpolator factor for the scanning positioner, then initialize and home the scanning group:

  PositionerHardInterpolatorFactorSet (Positioner, Factor) ; Maximum value 200

  GroupInitialize(Group) ; Initialize scanning group

  GroupHomeSearch(Group) ; Search home for the scanning

13.3.3.3.2 Scan execution

- If needed, set PCO pulse duration and pulse settling time:

  PositionerPositionComparePulseParametersSet(Positioner, PulseDuration,SetlingTime)

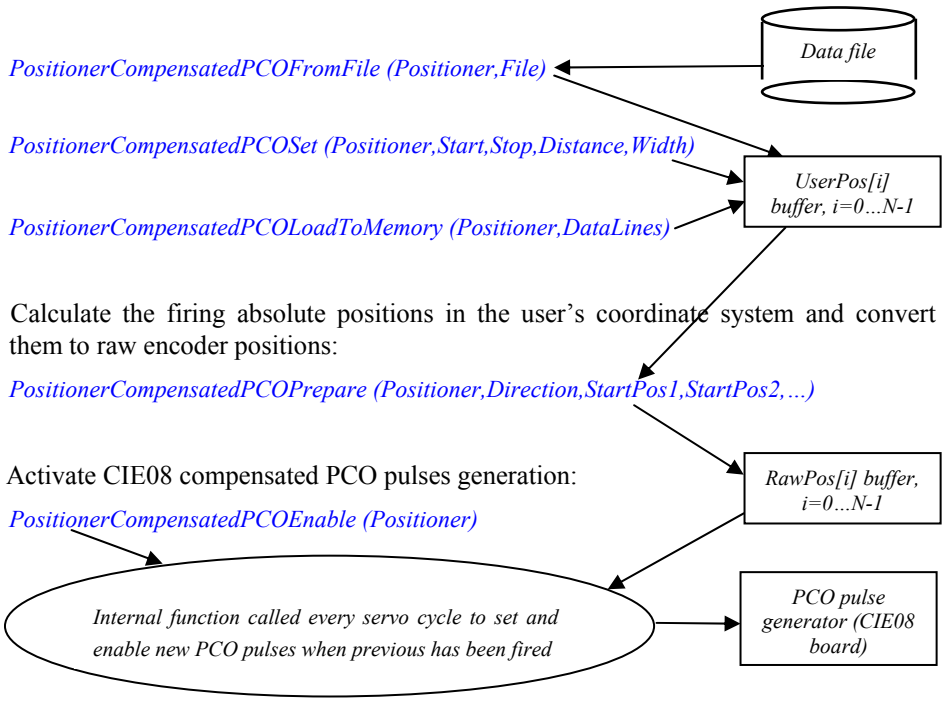
- Move the scanning group to the scan start position (*outside of the scanning zone*):

  GroupMoveAbsolute (Group, Position1, Position2, ...)

- Set the firing positions by reading data from file or loading to controller's memory or with a "set" function.

  Note that the firing positions defined with the following functions are only the offsets relative to the scanning positioner start position, that will be specified with the *PositionerCompensatedPCOPrepare()* function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



- Set motion parameters for scan:

  PositionerSGammaParametersSet(Positioner, ScanVelocity, ScanAcceleration, MinimumJerkTime,MaximumJerkTime)

- Move the scanning positioner across the scanning zone, during this move the firing pulses will be generated:

  GroupMoveRelative (Positioner, ScanDistance)

13.3.3.4 CIE08 compensated position compare related functions

Here is the list of the associated functions with a brief description. For detailed information, refer to the *XPS Programmer's Manual*.

13.3.3.4.1 Firing positions definition

There are three ways to enter the firing positions: reading from file, writing directly to the controller's memory or calculating with a "set" function.

Firing positions definition from a data file

Function *PositionerCompensatedPCOFromFile(Positioner, FileName)* reads firing positions from a data file to the controller's memory.

Firing positions definition from a "load to memory" function

Function *PositionerCompensatedPCOLoadToMemory(Positioner, DataLines)* appends firing positions to the controller's memory from *DataLines* parameter.

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

Firing positions definition from a "set" function

Function *PositionerCompensatedPCOSet(Positioner,Start,Stop,Distance,Width)* calculates a set of evenly spaced firing positions to the controller's memory.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

13.3.3.4.2 *Firing positions preparation*

Function *PositionerCompensatedPCOPrepare (Positioner, ScanDirection, StartPosition1, StartPosition2, ...)* calculates the firing at absolute positions, *in user's coordinate system* and converts them to firing absolute raw PCO positions, *in encoder's coordinate system*.

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

13.3.3.4.3 *Associated functions*

Pulses generation enable

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

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

Pulses generation abort

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

Pulses data reset

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

Pulses generation status get

The function *PositionerCompensatedPCOCurrentStatusGet (Positioner, Status)* gets the current status of CIE08 compensated PCO pulses generation.

13.3.4 Time Spaced Pulses (Time Flasher)

In the time spaced configuration, a first pulse is generated when the motion axis enters the time pulse window. From this first pulse, a new pulse is generated at every time interval until the positioner exits the time pulse window.

Hardware attains less than 50 ns jitter for the trigger pulses. The duration of the pulse is 200 nsec by default and can be modified using the function **PositionerPositionComparePulseParametersSet ()**. Possible values for the PCOPulseWidth are: 0.2 (default), 1, 2.5 and 10 (µs). Please note, that only the falling edge of the trigger pulse is precise and only this edge should be used for synchronization irrespective of the PCOPulseWidth setting. Note also, that the duration of the pulse detected by the electronics may be longer depending on the time constant of your RC circuit. Successive trigger pulses should have a minimum time lag equivalent to the PCOPulseWidth time multiplied by two.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following functions are used to generate time spaced pulses:

PositionerTimeFlasherSet

PositionerTimeFlasherGet

PositionerTimeFlasherEnable

PositionerTimeFlasherDisable

The function PositonerTimeFlasherSet() defines the position window and the time intervals for the trigger signals. It has four input parameters:

Position Name

Minimum Position

Maximum Position

Time Interval

The time interval must be greater than or equal to 0.0000004 seconds (0.4 µs) and less than or equal to 50 seconds. Furthermore, the time interval must be a multiple of 25 ns.

To enable the time spaced pulses, the function PositionerTimeFlasherEnable() must be sent.

Example 1

GroupInitialize(MyStage)

GroupHomeSearch(MyStage)

PositionerTimeFlasherSet(MyStage.X,5, 25, 0.00001)

PositonerTimeFlasherEnable(MyStage.X)

GroupMoveAbsolute(MyStage,30)

PositionerTimeFlasherDisable(MyStage.X)

The group has to be in a READY state for the time flasher to be enabled. Also, the PositionerTimeFlasherSet() function must be completed before the PositionerTimeFlasherEnable() function. In this example, one trigger pulse is generated every 0.00001 seconds or at a rate of 100 kHz between the minimum position of 5 mm and the maximum position of 25 mm. The first trigger pulse will be at 5 mm and the last trigger pulse will be at 25 mm or before.

The output pulses are accessible from the PCO connector at the back of the XPS controller, See appendix E, PCO connectors for details.

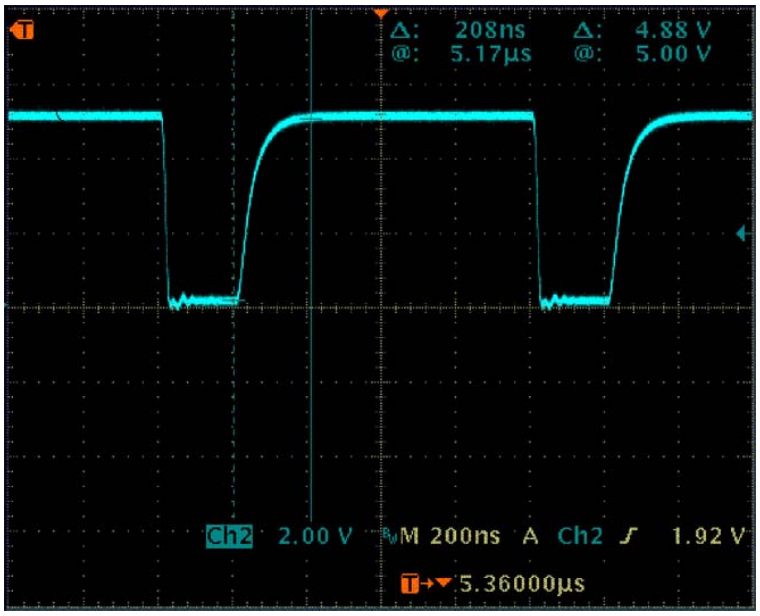


Figure 50: Temporal resolution of time spaced pulses in oscilloscope view.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example 2

The time flasher function is of particular use with high precision (direct drive) stages. At high speeds, these stages typically provide very good speed stability. In other words, the position change over a short time interval is highly consistent and repeatable. Hence, time spaced pulses can be used for synchronization with similar, in some cases even higher precision as distance spaced pulses. The time spaced pulse configuration, however, provides some further flexibility with regards to the nominal distance between successive triggers.
Consider an XM stage for instance. XM stages feature an analog encoder with 4 µm signal period. The max. resolution of the distance spaced pulses is 20 nm (setting PositionerHardInterpolatorFactorSet(200)). If the goal is to get pulses at a nominal distance of 268 nm at a speed of 200 mm/s speed, this is not possible using the distance spaced pulse configuration. Either 260 nm or 280 nm are possible, but not 268 nm. With some minor adjustments to the target speed, however, this is possible using the time spaced pulse configuration:

- The target speed is 200 mm/s, the desired distance between successive pulses is 268 nm. So the nominal time interval between successive pulses is:
  268 nm / 200 mm/s = 1.340 µs
- Round this nominal value to the next possible time interval, means to the next integer multiple of 25 ns: 1.350 µs
  Use this rounded time interval to calculate a corrected velocity:
  268 nm / 1.350 µs = 198.51852 mm/s

  GroupMoveAbsolute(MyStage.X, -50)
  PositionerSGammaParametersSet(MyStage.X, 198.51852, 2500, 0.02, 0.02)
  PositionerTimeFlasherSet(MyStage.X, -30, 30, 0.00000135)
  PositionerTimeFlasherEnable(MyStage.X)
  GroupMoveAbsolute(MyStage.X)
  PositionerTimeFlasherDisable(MyStage.X)

In this example, a first pulse is generated when the stage crosses the position -30 mm. Further pulses are generated every 1.350 µs until the stage reaches the maximum position of +30 mm. Since the stage moves at a speed of 198.51852 mm/s, the nominal distance between successive pulses is: 198.51852 mm/s * 1.35 µs = 268 nm.

13.3.5 AquadB Signals on PCO Connector

In the AquadB signals configuration, AquadB encoder signals are provided on the PCO connector, see Appendix E, PCO connector for details and pinning. These signals are either output always (Always configuration), or only when the positioner is within a defined position window (Windowed configuration).

When used with stages that feature a digital encoder (AquadB), the AquadB signals are the same as the encoder signals of the stage. When used with SinCos encoders (AnalogInterpolated), the resolution of the AquadB signal is defined by the signal period of the encoder and the settings of the hardware interpolator by the function PositionerHardInterpolatorFactorSet ().

Example

XM stages feature an analog encoder with a signal period of 4 µm. With the setting PositionerHardInterpolatorFactorSet (200) the post-quadrature resolution of the AquadB signals is: 4 µm / 200 = 0.02 µm. In this case one full period of the AquadB signals equals 0.08 µm.

The following functions are used to configure AquadB signals:

PositionerPositionCompareAquadBWindowedSet

PositionerPositionCompareAquadBWindowedGet

PositionerPositionCompareEnable

PositionerPositionCompareAquadBAlwaysEnable

PositionerPositionCompareDisable

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The function PositonerPositonCompareAquadBAlwaysEnable() has only one input parameter, the positioner name. When sent, AquadB signals are generated always. To disable this mode use the function PositionerPositionCompareDisable().

The function PositonerPositonCompareAquadBWindowedSet () has three input parameters.

Positioner name

Minimum Position

Maximum Position

To enable the AquadB signals, the function PositionerPositionCompareEnable() must be sent.

Example

GroupInitialize(MyStage)

GroupHomeSearch(MyStage)

PositionerPositionCompareAquadBWindowedSet(MyStage.X, 10, 20)

PositonerPositionCompareEnable(MyStage.X)

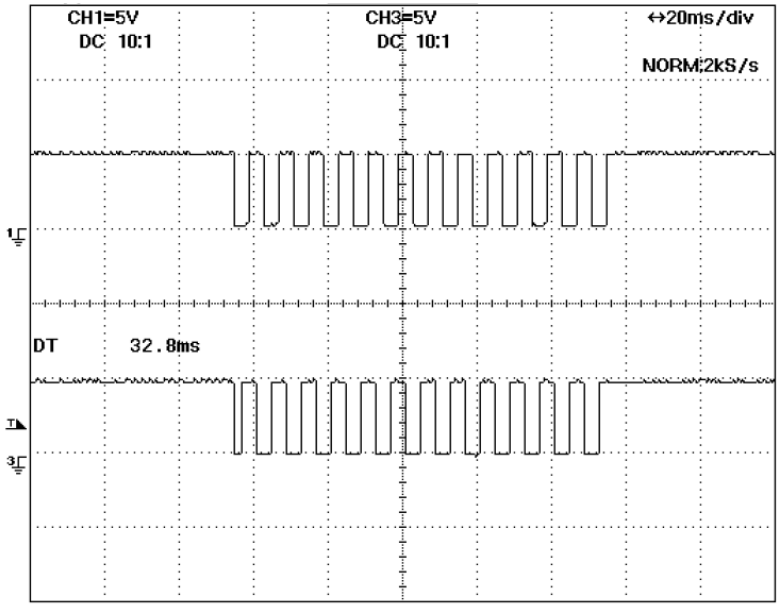
PositionerPositionCompareGet(MyStage, &MinimumPosition, &MaximumPosition, &EnableState)

This function returns the parameters previously defined, the minimum position 10, the maximum position 20 and the enabled state (1=enabled, 0 =disabled).

GroupMoveAbsolute(MyStage,30)

PositionerPositionCompareDisable(MyStage.X)

The figure below shows a screen shots from an oscilloscope for the example above.



The group has to be in a READY state for the position compare to be enabled. Also, the PositionerPositionCompareAquadBWindowedSet() function must be completed before the PositionerPositionCompareEnable() function. In this example, AquadB signals are generated when the positioner is between the minimum position of 10 mm and the maximum position of 20 mm.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE

The AquadB signal configuration is only available with positioners that have an encoder (AquadB or AnalogInterpolated).

The AquadB signals can not be provided at the same time as the distance spaced pulses (PCO) or the time spaced pulses.

The function PositionerPositionCompareEnable() enables always the last configuration sent, either distance spaced pulses defined with the function PositionerPositionCompareSet() or AquadB pulses defined with the function PositionerPositionCompareAquadBWindowedSet().

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

14.0 Control Loops

14.1 XPS Servo Loops

14.1.1 Servo structure and Basics

The XPS controller can be used to control a wide range of motion devices, which are categorized by the XPS as “positioners”. Within the structure of the XPS' firmware, a “positioner” is defined as an object with an associated profile (trajectory), a PID corrector, a motor interface, a driver, a stage and an encoder.

The general schematic of a positioner servo loop is below.

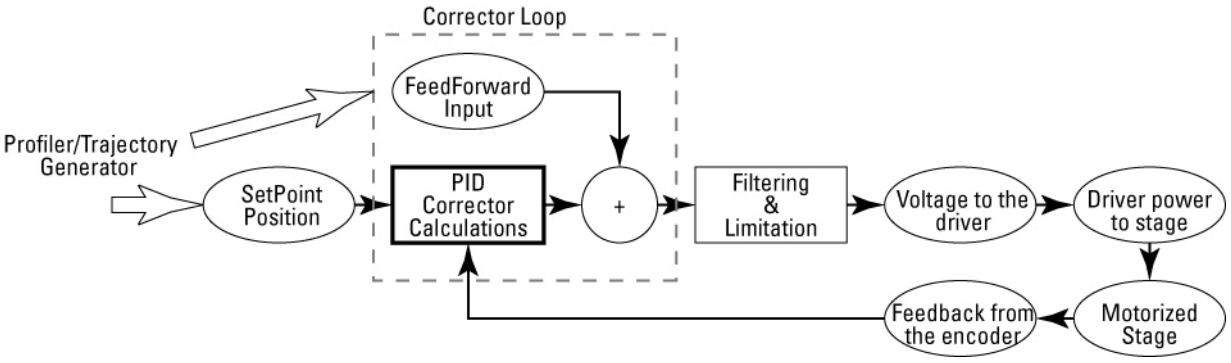


Figure 51: Servo structure and Basics.

The calculations done by the “servo loop” result in a voltage output from the controller that is applied to the driver, which can be either any of Newport's Universal drive modules or to an external driver through the XPS pass-through module. Depending on the corrector loop type selected, the level of this output voltage can be the result of two gain factors, the PID corrector and the FeedForward loop. The XPS has imbedded configuration files that provide optimized corrector loop settings for all Newport stages. Non-Newport stages may need to be assigned a specific corrector loop setting during the set-up process. In addition to the two main gain loops the XPS also adds filtering and error compensation parameters to this servo loop to improve system response and reliability.

The profiler (Trajectory Generator) within the controller calculates in real time, the position, velocity, and acceleration/deceleration that the positioner must follow to reach its commanded position (Setpoint Position). This profile is updated at a rate of 2.5 kHz.

The PID corrector then compares the SetpointPosition, as defined by the profiler, and the current position, as reported by the positioner's encoder, to determine the current following error. The PID corrector then outputs a value that the controller uses to maintain, increase or decrease the output voltage, which is applied to the driver. This loop is updated at a rate of 10 kHz. The adjustment of the PID parameters allows users to optimize the performance of their positioner or system by increasing or decreasing the responsiveness of the output to increasing or decreasing following errors. Refer to the section 14.3 on PID tuning for more information and tips on PID tuning. The PID corrector loop and trajectory generation loop rates have been optimized to provide the highest level of precision. In most applications the critical control loop is the PID corrector since it has the most significant impact on positioning performance. Because of this, the PID loop is updated 4 times (10/2.5) during each profiler cycle to improve profile execution and minimize following errors.

The Feed-Forward gain generates a voltage output to the driver that is directly proportional to the input. The purpose of this gain is to generate a movement of the positioner as close as possible to the desired move that is independent of the encoder feedback loop. Adding this Feed-Forward gain can help reduce any encountered following errors and thus requires less compensation by the PID gain corrector. For

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

example, if a driver and positioner respond to a constant voltage by moving at a constant speed, then feed forward input would be dictated by the SetpointSpeed.

The XPS stores standard Newport stage configuration files that can be used to quickly and easily develop the stage and system initialization (.ini) files. Below is an example of a typical stage and the type of DriverName, MotorDriverInterface and CorrectorType each is assigned. These standard Newport settings will be optimal for virtually every application and users would only need to modify their corrector loop parameters (Kp, Kd, Ki) to optimize positioner performance. Similar configurations can be adopted for non-Newport stages that are of similar motor driver types.

- Stages with high current (> 3 A) DC motor (RV, IMS) (with tachometer or back-emf estimation):

  DriverName: XPS-DRV01, 03

  ◇ ±10 V Input gives ±ScalingVelocity (stage velocity).

  ◇ Speed loop & Current loop configured by hardware.

  MotorDriverInterface: AnalogVelocity

  CorrectorType: PIDFFVelocity for Speed loop and PIDFFAcceleration for current loop.

- Stages with DC motor driven through a current loop (RGV) (no tachometer):

  DriverName: XPS-DRV02

  ◇ ±10 V Input gives ±ScalingAcceleration (stage acceleration).

  ◇ Current loop configured by hardware.

  MotorDriverInterface: AnalogAcceleration

  CorrectorType: PIDFFAcceleration

- Stages with low current (< 3 A) DC motor & tachometer (VP):

  DriverName: XPS-DRV01 in velocity mode.

  ◇ Input 1: ±10 V results in ±ScalingVelocity (theoretical stage velocity).

  ◇ Input 2: ±10 V results in ±ScalingCurrent (3 A).

  ◇ Speed loop programmable.

  MotorDriverInterface: AnalogVelocity

  CorrectorType: PIDFFVelocity

- Stages with low current (<3 A) DC motor, without tachometer (ILSCC type):

  DriverName: XPS-DRV01 in voltage mode.

  ◇ Input 1: ±10 V results in ±ScalingVoltage (48 V).

  ◇ Input 2: ±10 V results in ±ScalingCurrent (3 A).

  MotorDriverInterface: AnalogVoltage

  CorrectorType: PIDDualFFVoltage

- Stages with Stepper motor & Encoder (UTSPP, RVPE, ILSPP…):

  DriverName: XPS-DRV01 in stepper mode.

  ◇ Input 1: ±10 V results in ±ScalingCurrent in motor winding 1.

  ◇ Input 2: ±10 V results in ±ScalingCurrent in motor winding 2.

  MotorDriverInterface: AnalogStepperPosition

  CorrectorType: PIPosition

- Stages with Stepper motor & no encoder (TRA, SR50PP, PR50PP, MFAPP):

  DriverName: XPS-DRV01 in stepper mode.

  ◇ Input 1: ±10 V results in ±ScalingCurrent in motor winding 1.

  ◇ Input 2: ±10 V results in ±ScalingCurrent in motor winding 2.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MotorDriverInterface: AnalogStepperPosition

CorrectorType: NoEncoderPosition

These are just examples of available positioner associations in the XPS. The flexibility of positioner associations allows many other configurations to be developed to drive non-Newport positioners or other products. Before developing other configurations, the user must be aware that the main goal of creating these associations is to match the servo loop output to the appropriate driver input as stated by the manufacturer. For instance:

- The Corrector PIPosition is used when a constant voltage applied to a driver results in a constant position of the positioner (stepper motor, piezo, electrostrictive, etc.).
- Corrector PIDFFVelocity is used when a constant voltage applied to a driver results in a constant speed of the positioner (DC motor and driver board in speed loop mode).
- Corrector PIDFFAcceleration is used when a constant voltage applied to a driver results in a constant acceleration of the positioner (DC motor and driver board in current loop mode).
- Corrector PIDDualFFVoltage is used when a constant voltage applied to a driver results in a constant voltage applied to the motor (DC motor and driver board with direct PWM command).

14.1.2 XPS PIDFF Architecture

Corrector loops PIDFFVelocity, PIDFFAcceleration and PIDFFDualVoltage all use the same architecture as the PID corrector that is detailed below. PIPosition is a simplified version of this loop that is used to provide closed loop positioning via encoder feedback to stepper motor positioners.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

14.1.2.1 PID Corrector Architecture

The PID corrector uses the following error (SetpointPosition – EncoderPosition) as its input and applies the sum of three correction terms (Kp, Kd and Ki) to determine the output.

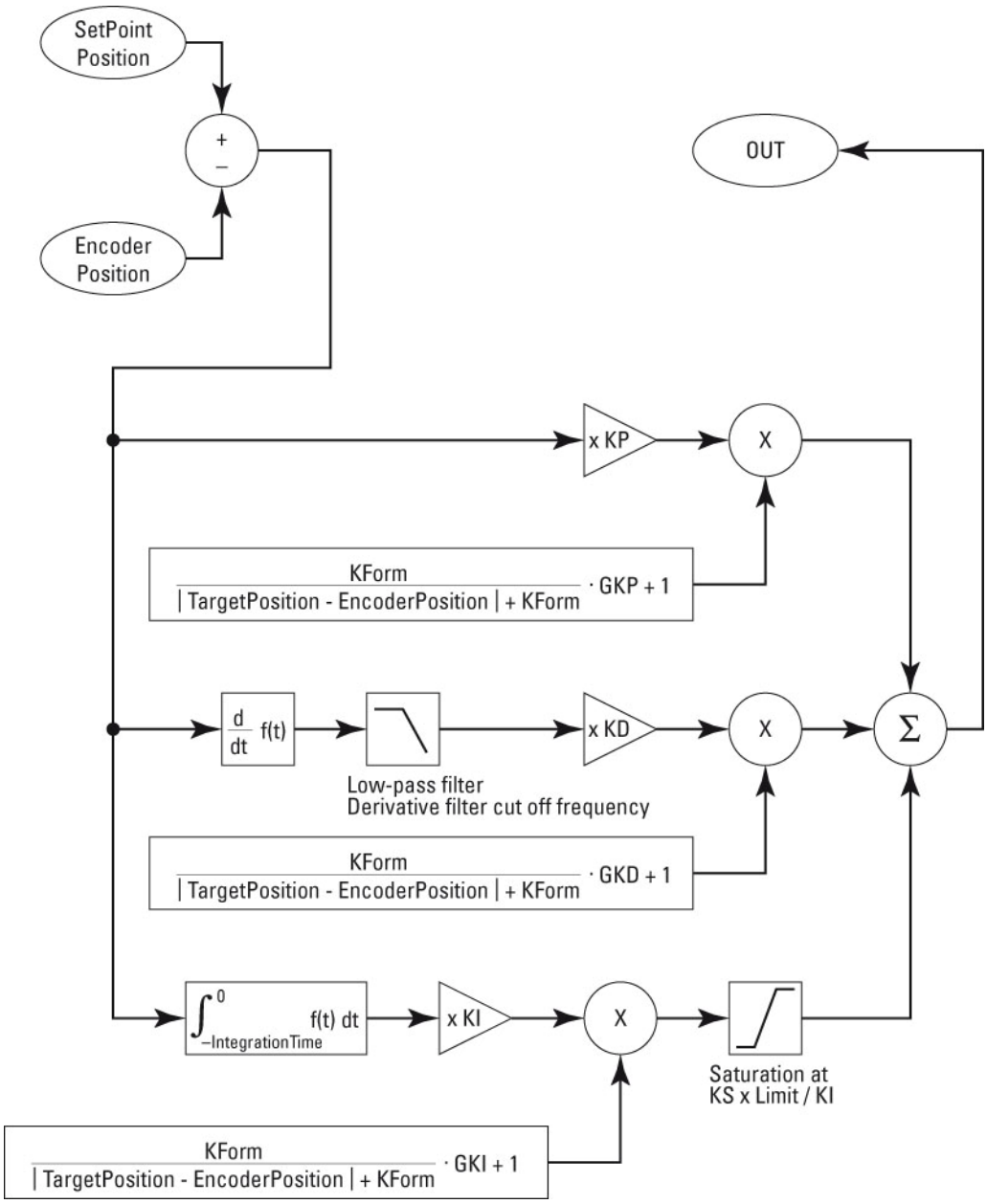


Figure 52: PID Corrector Architecture.

14.1.2.2 Proportional Term

The **Kp**, or proportional gain, multiplies the current following error of that servo cycle by the proportional gain value (Kp). The effect is to react immediately to the following error and attempt to correct it. Changes in position generally occur during commanded acceleration, deceleration, and in moves where velocity changes occur in the system dynamics during motion. As Kp is increased, the PID corrector will respond with a increased output and the error is more quickly corrected. For instance, if a positioner or group of positioners is expected to have small following errors, as is the case for small moves where overcoming static friction of the system is predominant, then the Kp may need to be increased to produce sufficient output to the driver. For larger moves, the following errors are generally larger and require lower Kp values to produce the desired output. Also note that for larger moves the kinetic friction of the system is generally

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

much lower than static friction and would generally require less correction gain than smaller moves. However, if Kp becomes too large, the mechanical system may begin to overshoot (encoder position > SetpointPosition), and at some point, it may begin to oscillate, becoming unstable if it does not have sufficient damping.

Kp cannot completely eliminate errors. However, since as the following error e, approaches zero, the proportional correction element, Kp x e, also approaches zero and results in some amount of steady-state error. For this reason other gain factors like Kd and Ki are required.

14.1.2.3 Derivative Term

The **Kd**, or derivative gain, multiplies the differential between the previous and current following error by the derivative gain value (Kd). The result of this gain is to stabilize the transient response of a system and can also be thought of as electronic damping of the Kp. The derivative acts as a gain that increases with the frequency of the variations of the following error:

$$\frac{d}{dt}=\left[\sin(2\pi \mathrm{Fr}\, t)\right]=2\pi \mathrm{Fr}\cos(2\pi \mathrm{Fr}\, t)$$

The result is that the derived term becomes dominant at high frequencies, compared to the proportional and integral terms. For the same reason, the value of Kd is in most cases limited by high frequency resonance of the mechanics. This is why a low pass filter (cut off frequency = DerivativeFilterCutOffFrequency) is implemented in the derivative branch to limit excitation at high frequencies. Increasing the value of Kd increases the stability of the system. The steady-state error, however, is unaffected since the derivative of the steady-state error is zero.

These two gains alone can provide stable positioning and motion for the system. However to eliminate the steady state errors, an additional gain value must be used.

14.1.2.4 Integral Term

The Integral term **Ki** acts as a gain that increases when the frequency of the variations of the following error decrease:

$$\int\left[\sin(2\pi \mathrm{Fr}\, t)\right]=\left(\frac{1}{2\pi \mathrm{Fr}}\right)\sin(2\pi \mathrm{Fr}\, t)$$

The result is that the integral term becomes dominant at low frequencies, compared to the proportional and derivative terms. The gain becomes infinite when frequency = 0. Even a very small following error will generate an infinite value of the integral term. The advantage of the integral term is that it will eliminate any steady-state following error. However, the disadvantage is that the integral term can reach values where the corrector is saturated causing the system to become unstable at the end of a move and cause the positioner to hunt or dither. To reduce this effect, two additional parameters are included in the PID corrector to help prevent these instabilities, Ks and Integration Time.

Ks

The saturation limit factor Ks permits users to limit the maximum value of Ki that is applied to the total PID corrector output. The Ks saturation limit can be set between 0 and 1, a typical setting is 0.5. As an example, at a setting of 0.5, the maximum output generated by the Ki term applied to the PID output would be 0.5 x the maximum set output. However, if the Ki gain factor output is less than 0.5 x the maximum set output, then the entire gain will be applied to the PID corrector. This maximum output is set within the section MotorDriverInterface in the stages.ini using the parameters AccelerationLimit, VelocityLimit or VoltageLimit. Refer to the Programmers manual for more information on this function.

Integration Time

The IntegrationTime is used to adjust the duration for integration of the residual errors. This can help in applications where large following errors can occur during motion. The

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

use of a small Integration Time value will limit the integration range to the latter parts of the move, avoiding the need of a large overshoot at the end of the move to clear the integrated following error value. The drawback is that the static error will be less compensated.

14.1.2.5 Variable Gains

In addition to the classical Kp, Ki, and Kd gain parameters, the XPS PID Corrector Loop also includes variable gain factors GKp, GKd, and GKi. These can be used to reduce settling time on systems that have nonlinear behavior or to tighten the control loop during the final segment of a move. For example, a positioner or stage with a high level of friction will have a response which is dependent on the size of the move: friction is negligible for a large move but becomes a predominant factor for small moves. For this reason, the required response of the system to reach the commanded position is not the same for small and large moves. The optimum value of PID parameters for small moves is very often higher than the optimum value for large moves. It is advantageous to modify PID settings depending on the move size. For users that do not need to make PID corrector adjustments (or prefer not to) benefit from the compensations provided by the variable gain correctors. This compensation is made automatically by the XPS variable gain corrector by applying a gain that is driven by the distance between the Target Position (position that must be reached at the end of the motion) and the Encoder Position. As shown in the figure below, when the distance to move completion is large, the total output gain from these parameters is fractional (the "Kform term" is fractional), but as the move size or distance to final position is small the Kform term approaches 1 and full GKx output is provided.

GKp = 10 Kp = 2
Target Position = 0
Encoder Position = -100 to 100

$$Kp_{(Kform,\ Encoder\ Position)} = \left[1 + GK \cdot \left(\frac{Kform}{|\ Target\ Position - Encoder\ Position\ | + Kform} \right) \right] \cdot Kp$$

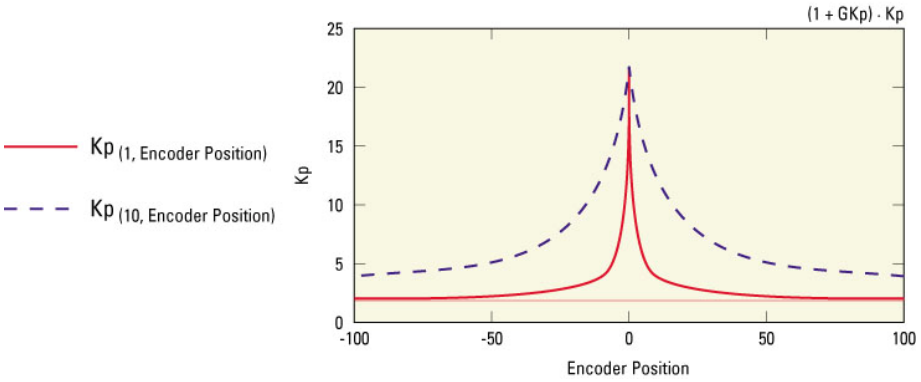


Figure 53: Variable Gains.

The parameter GKx is used to adjust the amplitude of the total output and the parameter Kform is used set how soon this Gkx is applied. As seen in the figure below, if a Kform of 1 is implemented, the GKx is not applied until the positioner is very close to its target position, in this case 0. But a Kform of 10 will implement the GKx much sooner and tighten the control of the loop further from the target position. This can be very effective when positioning high inertial loads or when very short settling times are critical. The default setting for the Kform parameter is 0 for all standard Newport stages.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

14.2 Filtering and Limitation

In addition to the various PID correctors and calculations, filtering and limitation parameters also have the same structure for all the correctors (PIDFFVelocity, PIDFFAcceleration and PIDFFDualVoltage, etc).

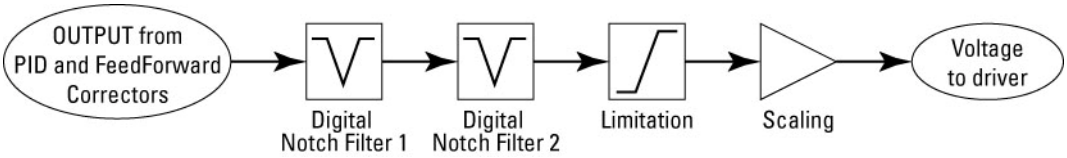


Figure 54: Filtering and Limitation.

The first section of the above diagram shows the succession of two digital notch filters. Each filter is defined by its central frequency (NotchFrequency), its bandwidth (NotchBandwidth) and its gain (NotchGain).

The gain, usually in the range of 0.01 to 0.1, is the value of the amplification of a signal at a frequency equal to the central frequency and the bandwidth is the range about the central frequency for which this gain is equal to a -3 db reduction.

Notch filters are typically used to avoid the instability of the servo loop due to the mechanic's natural frequencies, by lowering the gain at these frequencies. When they are implemented, these filters add some phase shift to the signal. This phase shift increases with the filter bandwidth and must remain small in the frequency range where the servo loop is active to maintain stability. The result is that notch filters are only effective at avoiding instabilities due to excessive and constant natural frequencies.

The last section of the diagram shows the limitation and scaling features. Scaling is used to transform units of position, speed or acceleration to a corresponding voltage. The Limitation factor is a safety that is used to limit the maximum voltage that can be applied to the driver to protect against any runaway or saturation situations that may occur.

14.3 Feed Forward Loops and Servo Tuning

14.3.1 Corrector = PIDFFVelocity

The PIDFFVelocity corrector should be implemented into applications where the positioner driver requires a "speed" input (constant voltage to the driver provides constant speed output to the positioner), using MotorDriverInterface = AnalogVelocity.

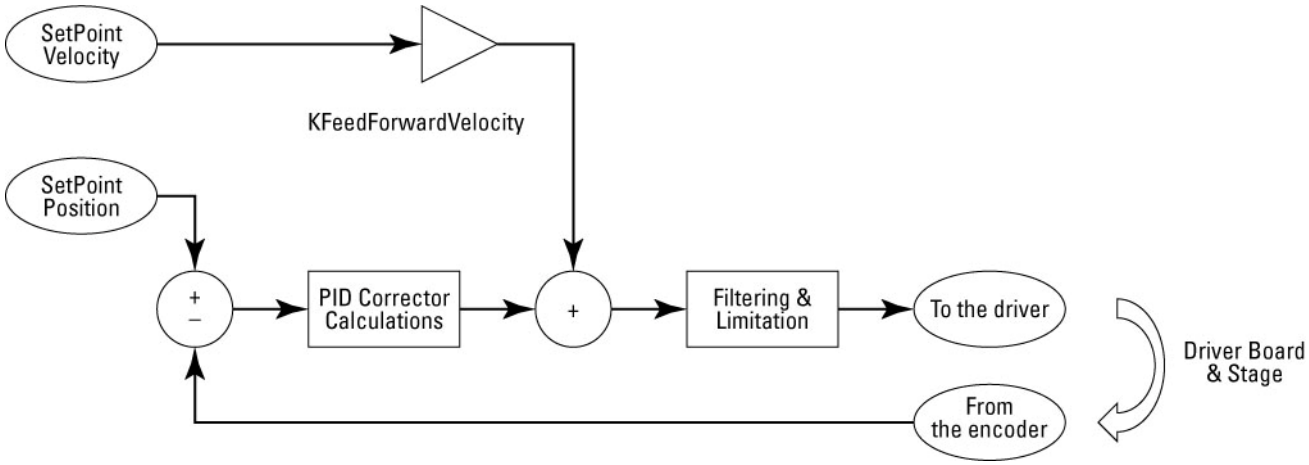


Figure 55: Corrector = PIDFFVelocity.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

14.3.1.1 Parameters

FeedForward Method:

- Velocity
- KFeedForwardVelocity is a gain that can be applied to this feed forward.
- When the system is used in open loop, the PID output is not applied and the feed forward gain is set to 1 (the entire output of the controller is FF gain).

PID corrector:

- Total output of the PID is a speed (units/s), so:

  Kp is given in 1/s.

  Ki is given in $1/s^2$.

  Kd has no unit.

Filtering and Limitation:

- ScalingVelocity (units/s) is the theoretical speed resulting from a 10 V input to the driver.
- VelocityLimit (units/s) is the maximum speed that can be commanded to the driver.

14.3.1.2 Basics

For a “perfect system” (no friction, all performance factors known, no following errors), a KFeedForwardVelocity value of 1 will generate the exact amount of output required to reach the TargetPosition.

The Kd parameter is generally redundant when using the speed loop of the driver and is usually set to zero, but a higher value can be used to improve the “tightness” of the speed loop.

The proportional gain Kp drives the cut-off frequency of the closed loop.

Due to the integration of the speed command in a position by the encoder, the overall gain of the proportional path at a given frequency Frq is equal to $Kp/2\pi Frq$. This gain is equal to 1 at Frq $P = Kp/2\pi$ (close to the cut-off frequency).

This frequency must remain lower than the cut-off frequency of the speed loop of the driver and lower than the mechanic’s natural frequencies to maintain stability.

The integral gain Ki drives the capability of the closed loop to overcome perturbations and to limit static error.

Due to the integration of the speed command in a position by the stage encoder, the overall gain of the integral path at a given frequency Frq is:

$$\text{Gain} = \frac{\text{Ki}}{(2 \cdot \pi \cdot \text{Frq})^2}$$

This gain is equal to one at FrqI:

$$\text{FrqI} = \frac{1}{2 \cdot \pi} \cdot \sqrt{\text{Ki}}$$

This frequency FrqI must typically remain lower than the frequency FrqP of the proportional path to keep the stability of the servo loop.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

14.3.1.3 Methodology of Tuning PID's for PIDFFVelocity Corrector (DC motors with or without tachometer)

1. Verify the speed in open loop (adjustment done using ScalingVelocity).
2. Close the loop, set Kp, increase it to minimize following errors to the level until oscillations/vibrations start during motion, then decrease Kp slightly to cancel these oscillations.
3. Set Ki, increase it to limit static errors and improve settling time until the appearance of overshoot or oscillation conditions. Then reduce Ki slightly to eliminate these oscillations.
4. Kd is generally not needed but it can help in certain cases to improve the response when the speed loop of the driver board is not efficient enough.

Note

To set the corrector parameters (loop type, Ki, Kp, Kd,...), use the following functions (refer to Programmer's Manual for details):

- **CorrectorType = PIDFFVelocity : PositionerCorrectorPIDFFVelocitySet(...)**
- **CorrectorType = PIDFFAcceleration: PositionerCorrectorPIDFFAccelerationSet(...)**
- **CorrectorType = PIDDualFFVoltage: PositionerCorrectorPIDDualFFVoltageSet(...)**
- **CorrectorType = PIPosition: PositionerCorrectorPIPositionSet(...)"**

14.3.2 Corrector = PIDFFAcceleration

The PIDFFAcceleration must be used in association with a driver having a torque input (constant voltage gives constant acceleration), using MotorDriverInterface = AnalogAcceleration. (AnalogSin60Acceleration, AnalogSin90Acceleration, AnalogSin120Acceleration, AnalogDualSin60Acceleration, AnalogDualSin90Acceleration or AnalogDualSin120Acceleration).

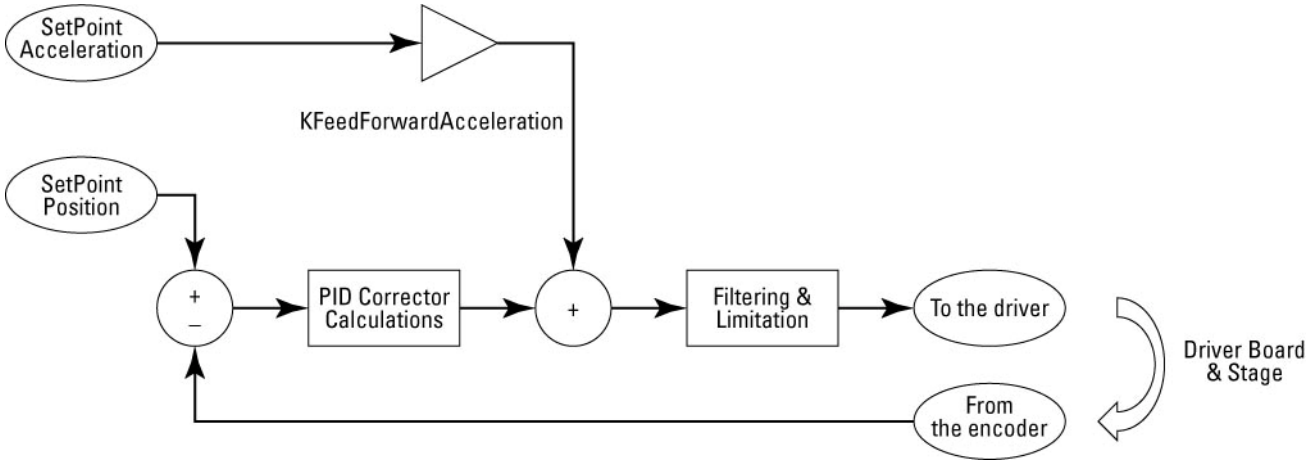


Figure 56: Corrector = PIDFFAcceleration.

14.3.2.1 Parameters

FeedForward method:

- A feed forward in acceleration is used.
- KFeedForwardAcceleration is a gain that can be applied to this feed forward.
- When the system is used in open loop, the PID output is cut and the feed forward gain is set to 1.

PID corrector:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Output of the PID is an acceleration value in units/s^2.

  Kp is given in 1/s^2.

  Ki is given in 1/s^3.

  Kd is given in 1/s.

Filtering and Limitation:

- ScalingAcceleration (units/s^2) is the theoretical acceleration of the stage resulting from a 10 V input to the driver (depends on the stage payload).
- AccelerationLimit (units/s^2) is the maximum acceleration allowed to be commanded to the driver.

14.3.2.2 Basics

The derivative term Kd drives the cut-off frequency of the closed loop and must be adjusted first (the loop will not be stable with only Kp).

Due to the double integration of the acceleration command in a position by the stage encoder, the overall gain of the derivative path at a given frequency Frq is equal to Kd/2πFrq. This gain is equal to one at FrqD = Kd/2π (close to servo loop cut-off frequency). This frequency must remain lower than the cut-off frequency of the current loop of the driver and lower to mechanical natural frequencies to keep the stability.

The proportional gain Kp drives mainly the capability of the closed loop to overcome perturbations at medium frequencies and to limit following errors. Due to the double integration of the acceleration command in a position by the stage encoder, the overall gain of the proportional part at a given frequency Frq is:

$$\text{Gain} = \frac{\text{Kp}}{(2 \cdot \pi \cdot \text{Frq})^2}$$

This gain is equal to one at FrqP:

$$\text{FrqP} = \frac{1}{2 \cdot \pi} \cdot \sqrt{\text{Kp}}$$

This frequency FrqP must remain lower than the frequency FrqD of the derivative part to keep the stability.

The integral gain Ki drives the capability of the closed loop to overcome perturbations at low frequencies and to limit static error.

Due to the double integration of the acceleration command in a position by the stage encoder, the overall gain of the integral part at a given frequency Frq is:

$$\text{Gain} = \frac{\text{Ki}}{(2 \cdot \pi \cdot \text{Frq})^3}$$

This gain is equal to one at FrqI:

$$\text{FrqI} = \frac{1}{2 \cdot \pi} \cdot \text{Ki}^{\frac{1}{3}}$$

This frequency FrqI must remain lower than the frequency FrqP of the proportional part to keep the stability.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

14.3.2.3 Methodology of Tuning PID's for PIDFFAcceleration Corrector (direct drive DC motors)

1. Verify the AccelerationFeedForward in open loop (adjustment done using ScalingAcceleration).

   Close the loop, set Kd, increase it to minimize following errors until vibrations appear during motion.
2. Decrease Kd to eliminate oscillations.
3. Set Kp, increase it to minimize following errors until the appearance of oscillations, decrease it to eliminate oscillations.
4. Set Ki, increase it to limit static errors and settling time until the appearance of overshoot/oscillations.

Note

To set the corrector parameters (loop type, Ki, Kp, Kd,...), use the following functions (refer to Programmer's Manual for details):

- **CorrectorType = PIDFFVelocity : PositionerCorrectorPIDFFVelocitySet(...)**
- **CorrectorType = PIDFFAcceleration: PositionerCorrectorPIDFFAccelerationSet(...)**
- **CorrectorType = PIDDualFFVoltage: PositionerCorrectorPIDDualFFVoltageSet(...)**
- **CorrectorType = PIPosition: PositionerCorrectorPIPositionSet(...)"**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

14.3.3 Corrector = PIDDual FFVoltage

The PIDDualFFVoltage must be used in association with a driver having a voltage input (constant voltage gives constant motor voltage), using MotorDriverInterface = AnalogVoltage.

Can also be used in velocity or acceleration command.

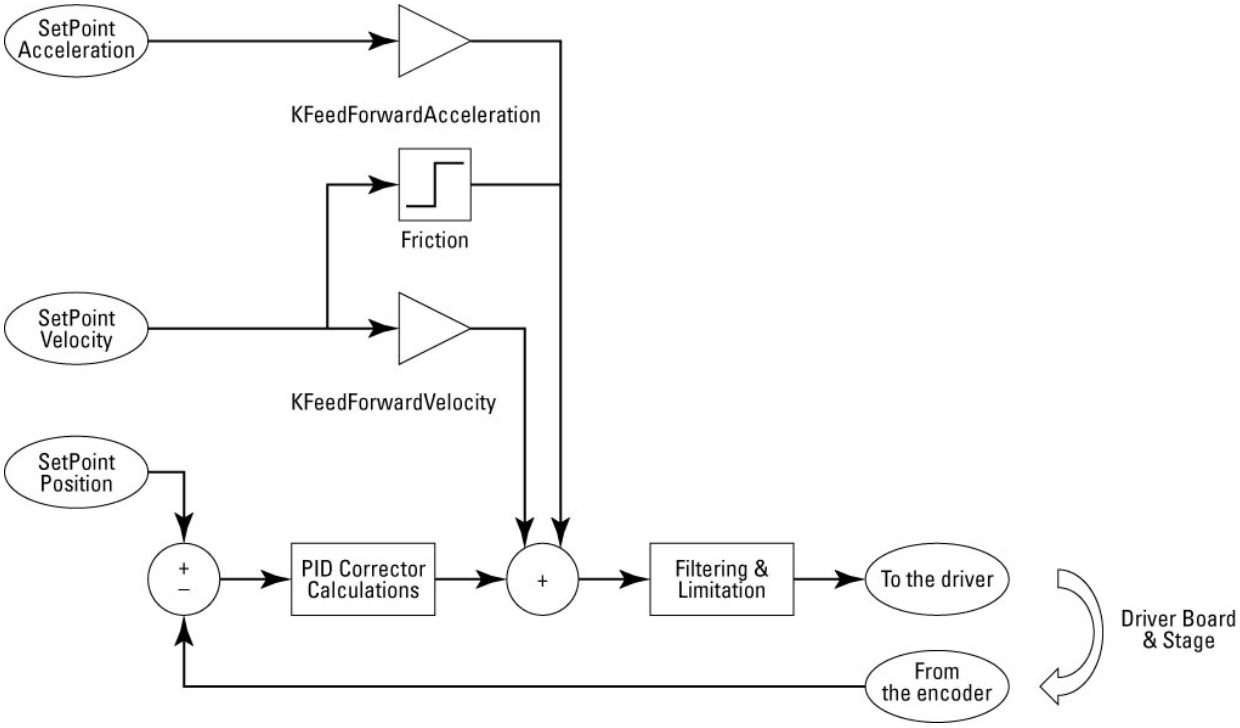


Figure 57: Corrector = PIDDual FFVoltage.

14.3.3.1 Parameters

FeedForward method:

- 3 feed forwards are used: Speed, Acceleration and Friction.
- KFeedForwardAcceleration is a gain that can be applied to the feed forward in acceleration.
- KFeedForwardVelocity is a gain that can be applied to the feed forward in velocity.
- Friction is a value which is applied with the sign of the velocity.
- When the system is used in open loop, the PID output is cut and only one feed forward in velocity is applied with the gain defined by KFeedForwardVelocityOpenLoop.

PID corrector:

- Output of the PID is a voltage.

  Kp is given in V/unit.

  Ki is given in V/unit/s.

  Kd is given in V/s/unit.

Filtering and Limitation:

- ScalingVoltage is the theoretical motor voltage resulting from a 10 V input on the driver (48 V).
- VoltageLimit (volts) is the maximum motor voltage allowed to be commanded to the driver.

Refer to the XPS-Q8 Configuration Wizard Document for a detailed explanation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

14.3.3.2 Basics

The PIDDualFFVoltage corrector can be seen as a mix between the PIDFFVelocity and PIDFFAcceleration correctors. It is difficult to give a precise picture of this behavior which depends a lot on the response of the stage (speed and acceleration versus motor voltage).

14.3.3.3 Methodology of Tuning PID's for PIDDualFF Corrector (DC motors with tachometers)

1. Adjust KFeedForwardVelocityOpenLoop to optimize the fidelity of the speed at high speed.
2. Close the loop using the same value for KFeedForwardVelocity, set Kp, increase it to minimize following errors until oscillations/vibrations appears during motion, decrease Kp to eliminate oscillations.
3. Set Kd, increase until oscillations/vibrations appear during motion, and decrease it to eliminate oscillations.
4. Increase Ki to cancel static error and minimize settling time until appearance of overshoot/oscillations.

14.3.4 Corrector = PIPosition

PIPosition corrector can be used with AnalogStepperPosition or AnalogPosition interface.

The AnalogPosition interface is to be used with a driver having a position input (example = piezo driver).

The AnalogStepperPosition interface is to be used with a driver having two sine and cosine current inputs (constant voltage gives constant currents in motor windings so position is constant).

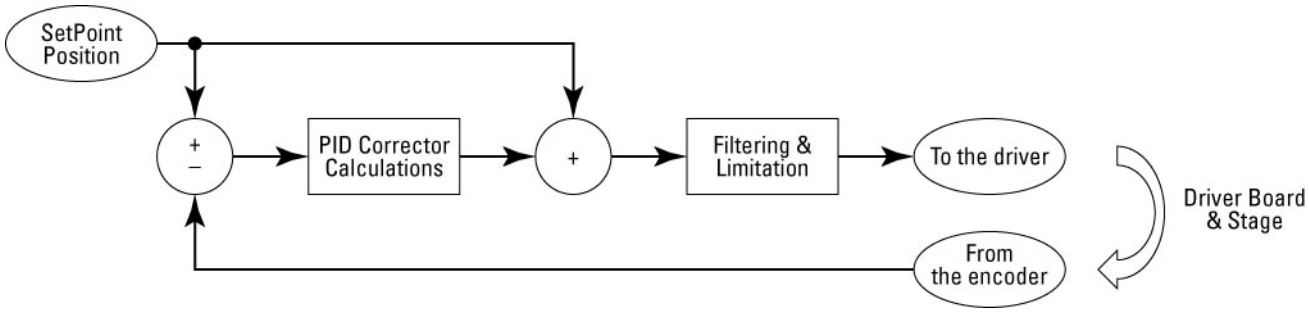


Figure 58: Corrector = PIPosition.

14.3.4.1 Parameters

FeedForward:

- One feed forward in position. No adjustable gain.
- When the system is used in open loop, the PI output is cut and the feed forward in position is applied.

PI corrector:

- Output of the PI is a position.

  Kp has no units.

  Ki is given in 1/s.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

14.3.4.2 Basics & Tuning

In most cases, only Ki is needed to correct static errors.

The overall gain of the integral part of the servo loop at a given frequency Frq is:

$$\text{Gain} = \frac{\text{Ki}}{2 \cdot \pi \cdot \text{Frq}}$$

This gain is equal to one at:

$$\text{FrqI} = \frac{\text{Ki}}{2 \cdot \pi}$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

15.0 Analog Encoder Calibration

This section refers only to analog sine encoder inputs. The purpose of the analog encoder interpolation feature is to improve the stage accuracy by detecting and correcting analog encoder errors such as offsets, sine to cosine amplitude differences, and phase shift.

Other kinds of errors can exist in the encoder such as impure sine or cosine signals. This feature will not compensate for them and will disturb the results of the calibration process.

Also, this calibration process assumes that the errors are small, i.e., less than a few percent.

Below are figures and numbers to illustrate the type of errors and their impact on accuracy.

Offset Error

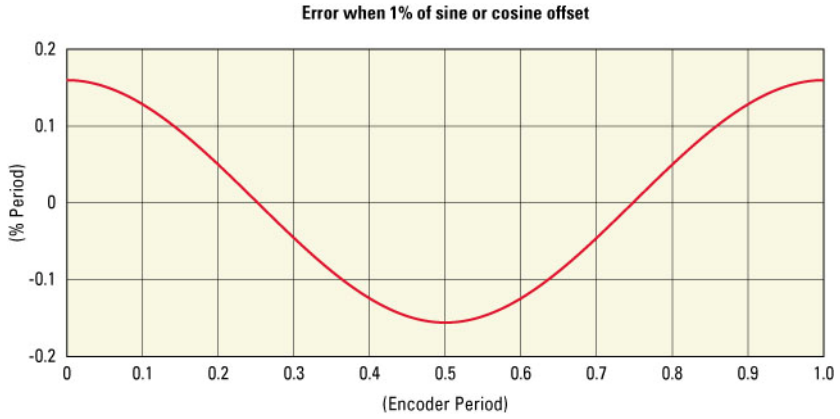


Figure 59: Offset Error.

The offset error generates 0.32% interpolation error per percent offset on the sine or cosine signals. With a 20 µm scale pitch, 1% sine offset generates 63.5 nm peak to peak interpolation error.

Note

The real signal is not always symmetrical to 0. The offset error is defined as the difference between the signal's horizontal axis where it is symmetrical and 0.

Amplitude Mismatch

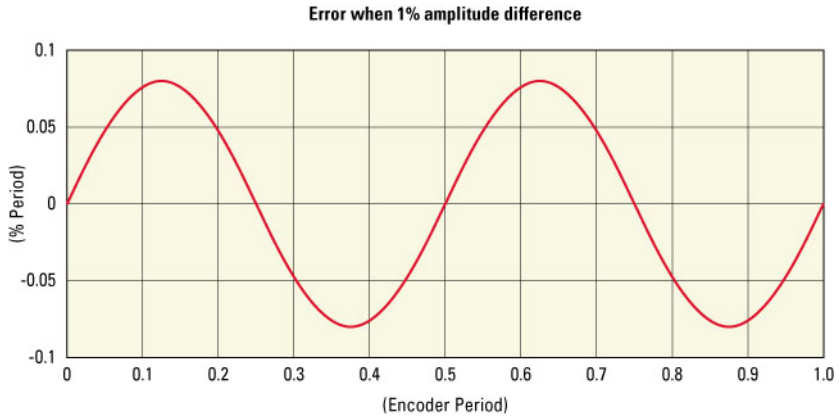


Figure 60: Amplitude Mismatch.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The amplitude mismatch between sine and cosine signals generates 0.17% interpolation error per percent amplitude mismatch. With a 20 µm scale pitch, 1% amplitude mismatch generates 33 nm peak to peak interpolation error.

Note

Positive amplitude is the distance between the signal's maximum value and the signal axis. Negative amplitude is the distance between the signal's minimum value and the signal axis. If the positive amplitude and negative amplitude are not equal, there is amplitude mismatch.

Phase Shift

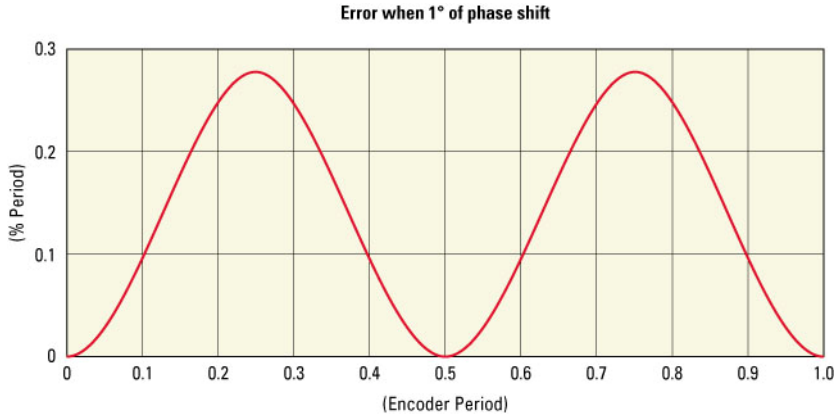


Figure 61: Phase Shift.

The phase shift between sine and cosine generates 0.28% interpolation error per degree phase shift. With a 20 µm scale pitch, 1 degree phase shift between sine and cosine generates 55.5 nm peak to peak error.

Combined Errors

The combination of these errors is not a simple sum but is more likely a root mean square relationship. With a 20 µm scale pitch, 1% sine offset, 1% cosine offset, 1% phase mismatch and 1 degree phase error between sine and cosine generates 132.5 nm peak to peak error.

$$2000\sqrt{(0.32\%)^2+(0.32\%)^2+(0.164\%)^2+(0.28\%)^2}=111.373$$

Note that the calculated value, 111.373 nm is lower than the measured 132.5 nm.

Analog encoder compensation feature

The compensation for repeatable distortions of the analog encoder input signals is always active. It uses the following parameters read from the stages.ini file. The default values are 0 for all stages:

EncoderSinusOffset = 0 volts

EncoderCosinusOffset = 0 volts

EncoderDifferentialGain = 0

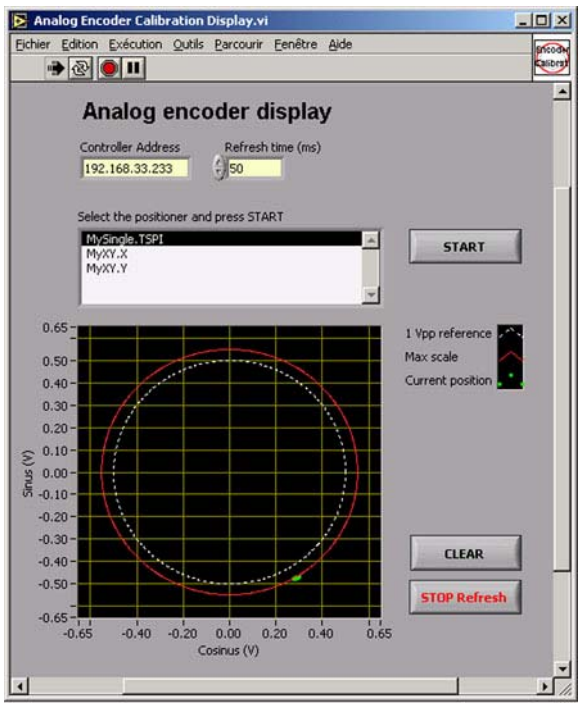
EncoderPhaseCompensation = 0 deg

The function **GroupInitializeWithEncoderCalibration()** initializes the positioner and runs the encoder calibration process. During calibration, the stage moves for 25 EncoderScalePitch and the controller determines the appropriate calibration values. The controller though, will not automatically apply these values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The function **PositionerEncoderCalibrationParametersGet()** returns the results of the last encoder calibration. To apply these values, add them manually to the appropriate section in the stages.ini file, and reboot the controller.

In the folder ..\Admin\Public\Drivers\LabView\XPS-Q8 of the XPS controller, embedded in Examples.llb, there is a LabVIEW application to display the current analog encoder values. The display zone matches the maximum possible amplitude of the analog signals. When they are larger than this, the AD converter will clip and the interpolation error will increase dramatically. The dotted circle represents the 1 volt peak to peak "ideal" encoder, the red circle represents the current mean encoder settings and the green dot the current encoder value. This application uses the function PositionerEncoderAmplitudeValuesGet() for display.



Example of the use of the functions

GroupInitializeWithEncoderCalibration(MyGroup)

PositionerEncoderCalibrationParametersGet(MyGroup.MyStage)

This function returns the encoder calibration parameter values: encoder sine signal offset, encoder cosine signal offset, encoder differential gain, and encoder phase compensation. These values need to be entered in the appropriate section of the stages.ini.

PositionerEncoderAmplitudeValuesGet(MyGroup.MyStage)

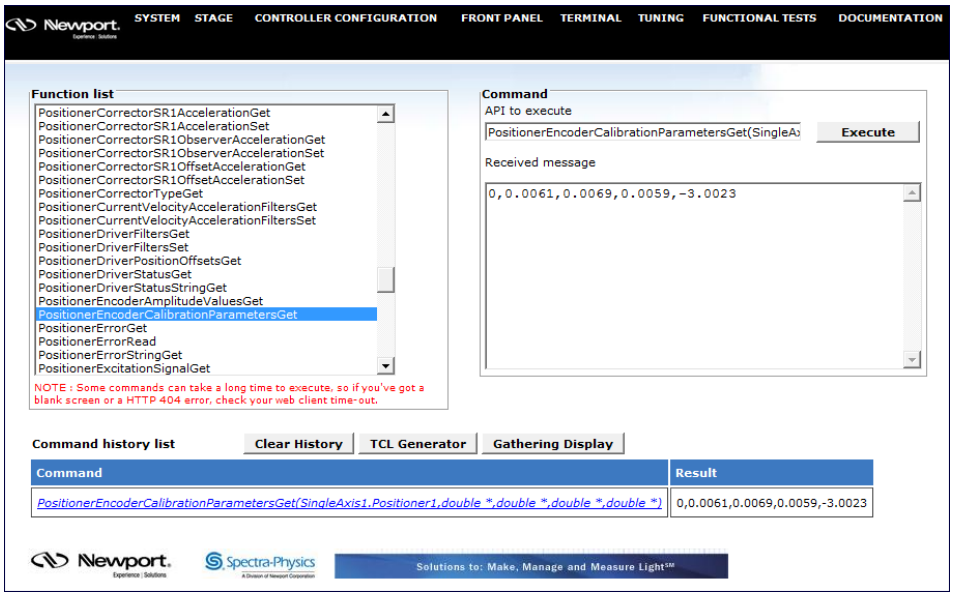
This function returns the encoder amplitude values: encoder sine signal maximum amplitude value, encoder sine signal current amplitude value, encoder cosine signal maximum amplitude value and encoder cosine signal current amplitude value.

Following is the complete process for calibrating a stage with an analog encoder interface:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

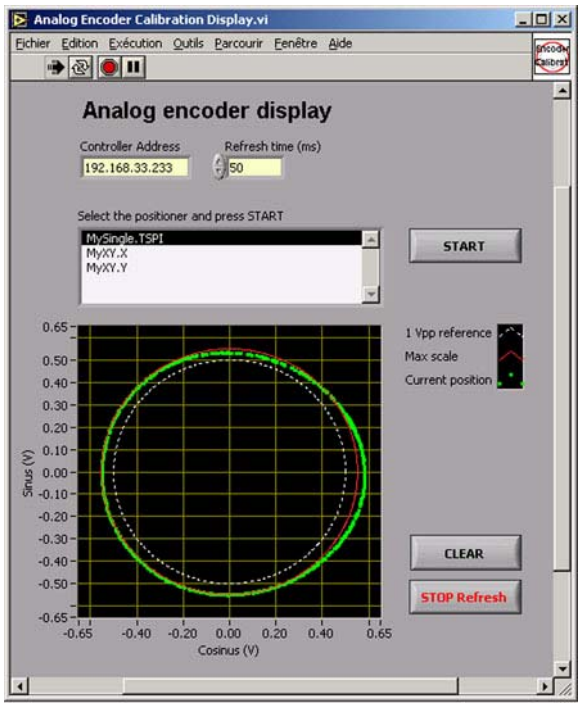
Step 1

Initialize the positioner and run the calibration routine.



Step 2

Start the AnalogEncoderCalibrationDiplay VI which is found in the ftp site. Move the positioner at very low speed.



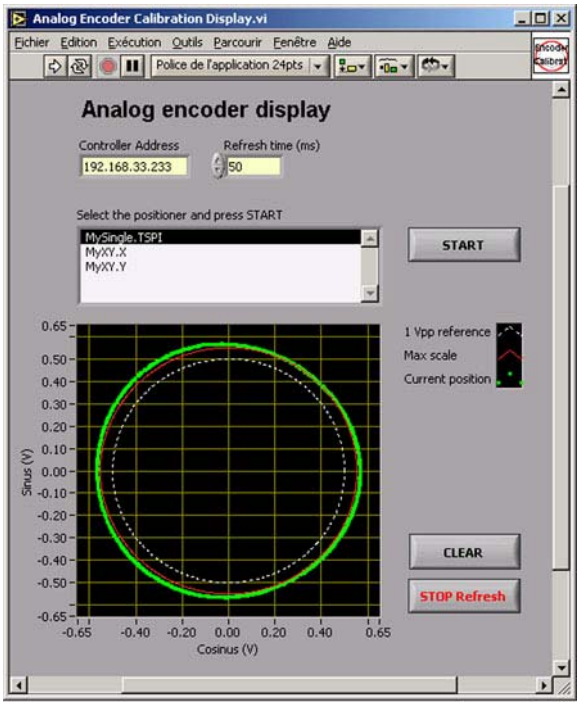
Notice the variations between the actual (green) values and the ideal (red) values. In this case, it makes sense to apply new compensation values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Step 3

Apply the compensation values gathered in step 1 into the stages.ini; reboot the controller.

Initialize the positioner: run the AnalogEncoderCalibrationDiplay VI; move the positioner at a very low speed.



Notice the difference to the previous results. It might be necessary to run the compensation at several positions and several times to optimize the results.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

16.0 Excitation Signal

16.1 Introduction

The excitation-signal function generates a typical signal (a sine, a blank noise or an echelon signal) that the controller sends to motors to excite the system. In measuring the output signal of the excited system, we can determine some system characteristics, such as the system transfer function.

NOTE

The excitation-signal function is allowed only with XPS-Q8 Precision Platform controller.

It's possible to generate an excitation signal for a gantry system : sending the same excitation signal to both primary and secondary positioners.

16.2 How to Use the Excitation-Signal Function

The PID excitation-signal function is only available with the stages controlled in acceleration (acceleration control, ex: brushless / linear motors), velocity (velocity control) or in voltage (voltage control). It is not used with the stages controlled in position (ex: stepper motors).

The excitation-signal function **PositionerExcitationSignalSet** can be executed only when the positioner is in the "READY" state. When the excitation-signal function is in process, the positioner is in the "ExcitationSignal" state. At the end of the process, the positioner returns to the "READY" state (see group state diagram).

This function sends an excitation command to the motor over a time period. This function is allowed for "PIDFFAcceleration", "PIDFFVelocity" or "PIDDualFFVoltage" control loop. The parameters to configure are *signal type* (0:sine, 1:echelon,2:random-amplitude,3:random-pulse-width binary-amplitude, integer), *frequency* (Hz, double), *amplitude* (acceleration, velocity or voltage unit, double) and *during time* (seconds, double).

The effective functional parameters for each mode are: (Limit means AccelerationLimit, VelocityLimit or VoltageLimit) :

– Sine signal mode : *Frequency* (>=1 and <= 5000), *Amplitude* (>0 and <= Limit), *Time*(>0)

– Echelon signal mode : *Amplitude* (>0 and <= Limit, or <0 and >= -Limit), *Time* (>0).

+ During *Time* : Signal = *Amplitude*

+ End of *Time* : Signal = 0

- Random-amplitude signal mode : *Amplitude* (>0 and <= Limit), *Time*(>0), *Frequency* (>= 1 and <= 5000).

  The signal is generated with a random value at every controller base time (Tbase = 0.1 ms), then is filtered with a second order low-pass filter at the cut-off *Frequency* value.

o Random-pulse-width binary-amplitude signal mode :

  Amplitude (>0 and ≤ Limit), *Time* (>0), *Frequency* (≥1 and ≤5000).

  The signal is a sequence of pulses (Signal = *Amplitude* or = 0) with the pulse randomly varied in width (multiple of Tbase).

  Frequency is the controlled system band-width (*cut-off frequency*), necessary for the PRBS (*Pseudo Random Binary Sequence*) function configuration.

The non-effective functional parameters can accept any value, the value 0 is recommended for simplicity.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

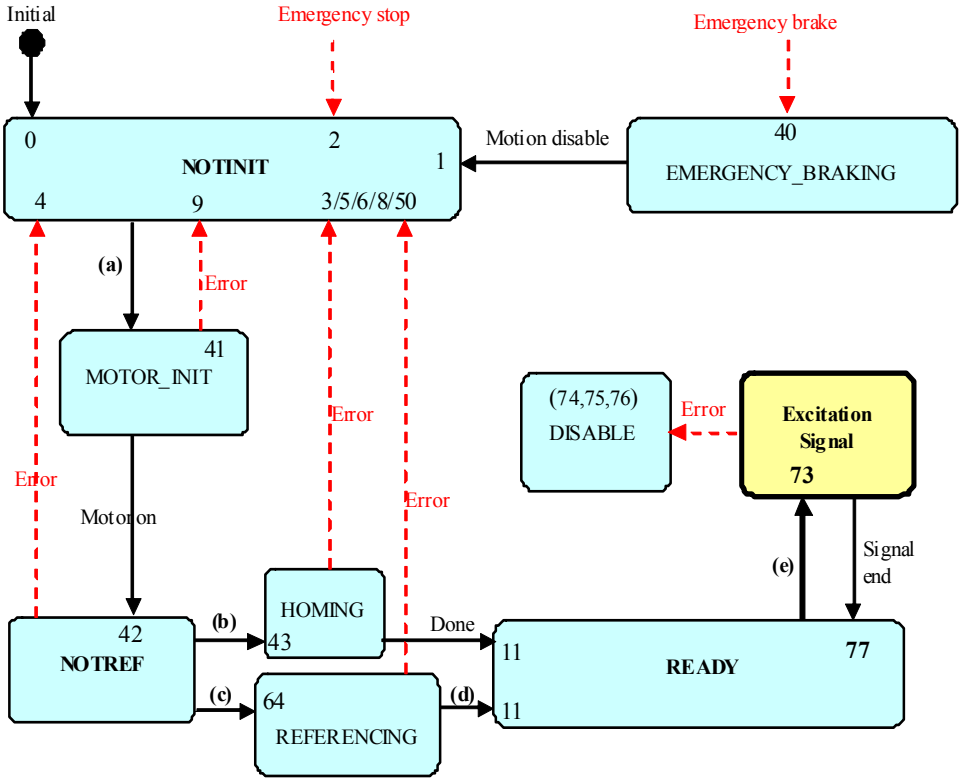
The *PositionerExcitationSignalGet()* function is used to get the parameters previously used with the *PositionerExcitationSignalSet()* function.

16.3 Group State Diagram

Notes :

The numbers in the boxes represent the values of the group status.

Bold transitions are driven by function, the others are internal transitions.



(a) GroupInitialize

(b) GroupHomeSearch or **(c)** GroupReferencingStart **(d)** GroupReferencingStop

(e) PositionerExcitationSignalSet

16.4 Function Description

- ***PositionerExcitationSignalGet***
- ***PositionerExcitationSignalSet***

(see ProgrammerManual.pdf)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

17.0 Pre-Corrector Excitation Signal

17.1 Description

The XPS firmware integrates functions to measure the response of the system by injecting different excitation signals inside the control loop.

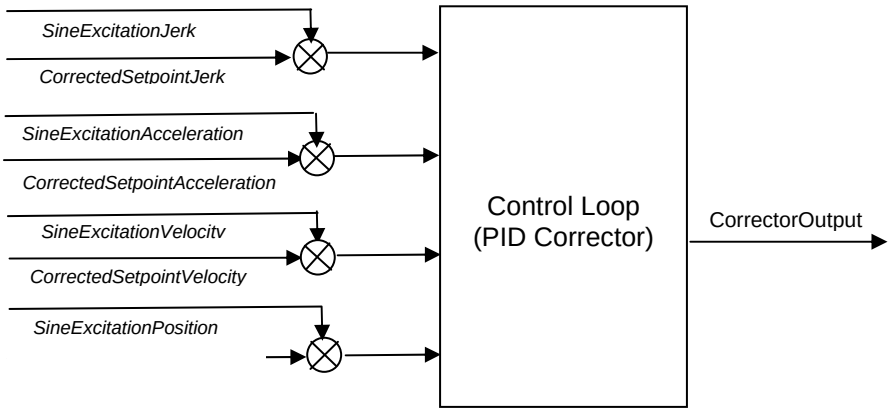
To extend the capabilities to use alternate ways measuring a system transfer function the injection of an excitation sine wave signal will be made available outside the control loop.

To benefit from the efficiency of the control loop using the feed-forward gains, the position, the velocity, the acceleration and the jerk will be injected in the control loop.

This new feature, coupled with the gathering capabilities of the controller will allow to inject the sine position excitation and concurrently gather the commanded position and the feed-back positions.

These gathered information can be stored in the controller and retrieved for further analysis.

The architecture of the implementation is the following:



This feature is allowed with "PIDFFAcceleration", "PIDFFVelocity" and "PIDDualFFVoltage" control loops.

This makes it available for stages controlled with acceleration for drivers integrating a current loop (such as for brushless, linear motors or D.C. motors), velocity for drivers integrating a velocity control loop (such as for D.C. motors with a tachometer) or with voltage for power amplifiers without any embedded control loop.

Refer to the XPS user's Manuals for more details on the different control loops.

The new SineExcitation are calculated every servo cycle based on the input parameters. Depending on the setting of the controller the servo cycle is up to 10 kHz.

17.2 Pre-corrector excitation signal wave forms

The exact forms of pre-corrector excitation signal of position, velocity, acceleration and jerk are the followings:

$\omega = 2\pi F$ *(F: excitation frequency)*

$ExcitationPosition = A*\cos(\omega t) - A$ *(A: excitation amplitude, t: current time)*

$ExcitationVelocity = (-A\omega)*\sin(\omega t)$

$ExcitationAcceleration = (-A\omega^2)*\cos(\omega t)$

$ExcitationJerk = (A\omega^3)*\sin(\omega t)$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

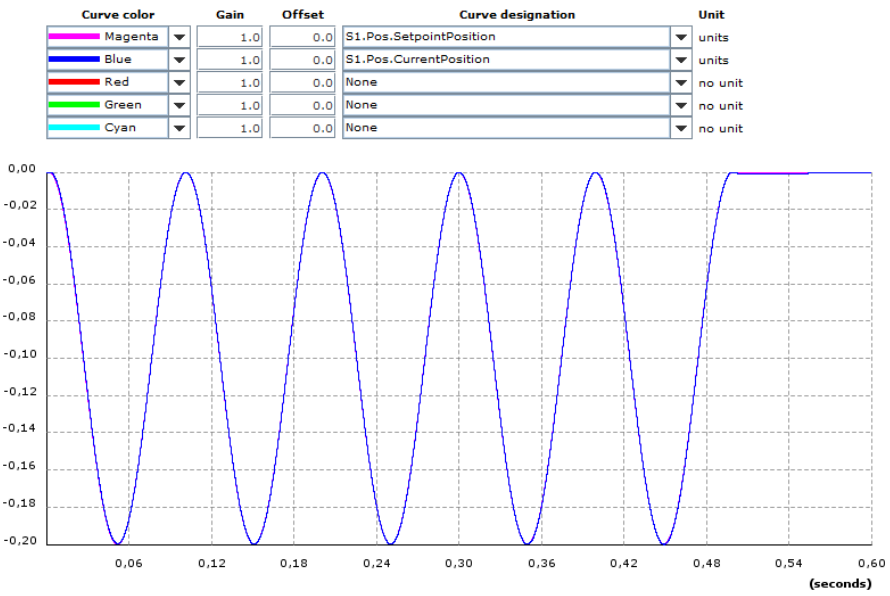


Fig. 1: Position excitation signals, amplitude A (unit).

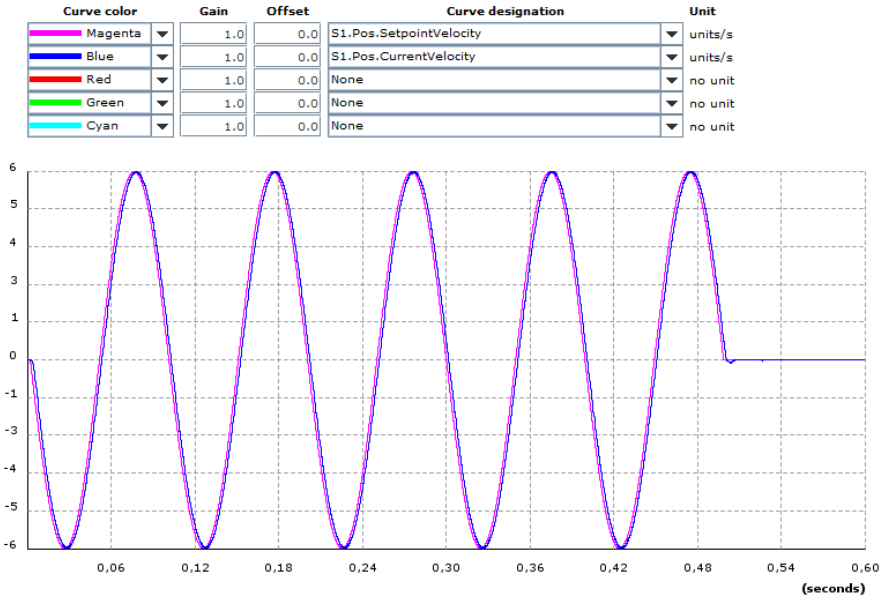


*Fig. 2: Velocity excitation signals, amplitude = A*2πF (units/s).*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

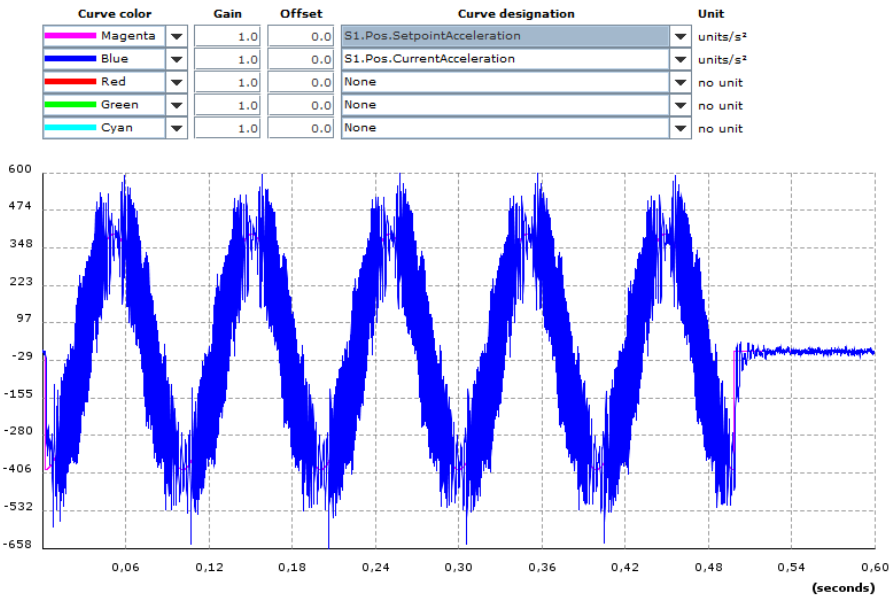


Fig. 3: Acceleration excitation signals, amplitude = A(2πF)^2 (units/s²).*

17.3 Technical Implementation

17.3.1 Use case

The following sequence of commands will allow to gather the required informations:

Prior to the use of this new feature the group must be in the ready state, to do so the following functions are required:

GroupInitialize()

GroupHomeSearch()

To gather the data for a given frequency, the following steps are required:

GatheringConfigurationSet()

EventExtendedConfigurationTriggerSet()

EventExtendedConfigurationActionSet()

EventExtendedStart()

PositionerPreCorrectorExcitationSignalSet()

GatheringStopAndSave()

To retrieve the gathered data different methods can be used, ftp transfer or programmatically using one of the following functions:

GatheringConfigurationGet()

GatheringCurrentNumberGet()

GatheringDataGet()

GatheringDataMultipleLinesGet()

GatheringStop()

GatheringRunAppend()

Refer to the XPS user's manaual for more details on the above mentionned listed functions.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The difference between the existing excitation signal function (*PositionerExcitationSignalSet*) and the new one (*PositionerPreCorrectorExcitationSignalSet*) is :

– *PositionerExcitationSignalSet* inserts the excitation signal into corrector output after the (*PID corrector*) control loop calculation.

– *PositionerPreCorrectorExcitationSignalSet* inserts the excitation into corrector inputs before the (*PID corrector*) control loop calculation.

17.3.2 Implementation

The *PositionerPreCorrectorExcitationSignalSet* function is implemented to inject the excitation signal

The excitation signal function *PositionerPreCorrectorExcitationSignalGet* can be executed only when the positioner is in the "READY" state. When the excitation signal process is in progression, the positioner is in the "ExcitationSignal" state. At the end of the process, the positioner returns to the "READY" state (see the group state diagram below).

The function parameters are:

– Frequency *(>= 0.1 and <= 0.5 / CorrectorISRPeriod (5000 Hz if CorrectorISRPeriod = 0.1µs)*)

– Amplitude *(> 0, amplitude of sine excitation signal , in unit of position)*

– Time *(> 0, seconds)*

The verifications of position amplitude (*SetpointPosition <= MaximumTargetPosition and (SetpointPosition – 2*Amplitude) >= MinimumTargetPosition*), velocity amplitude (*> 0 and <= MaximumVelocity*) and acceleration amplitude (*> 0 and <= MaximumAcceleration*) must be done in this function.

The *PositionerPreCorrectorExcitationSignalGet()* function is used to get the parameters previously set with the *PositionerPreCorrectorExcitationSignalSet()* function.

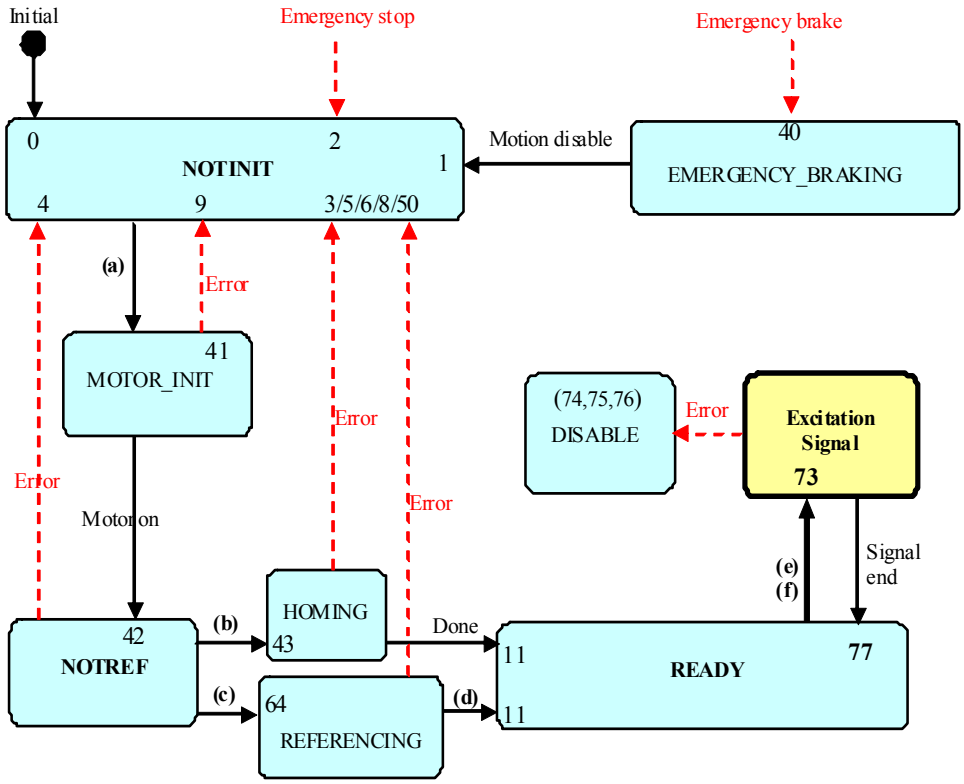
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

17.3.3 Group capsule state diagram modification

Notes

The numbers in the boxes represent the values of the group status.

Bold transitions are driven by function, the others are internal transitions.



(a) GroupInitialize
(b) GroupHomeSearch or **(c)** GroupReferencingStart **(d)** GroupReferencingStop
(e) PositionerExcitationSignalSet
(f) PositionerPreCorrectorExcitationSignalSet

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

18.0 Introduction to XPS Programming

For advanced applications and repeating tasks, it is usually better to sequence different functions in a program rather than executing them manually via the web site interface. Motion process programs can be written in different ways, but essentially are distinguished between host-managed and XPS-managed processes. A host-managed uses the Ethernet TCP/IP interface from a PC to control the XPS. The XPS-managed process is controlled directly by the XPS controller via a TCL script.

The chapter provides a brief introduction of the different ways of programming the XPS. This section, however, cannot address all details. For further information, refer to the TCL and the software drivers manual of the XPS controller which are accessible via the XPS web site.

Host-managed processes

Host-managed processes are recommended for applications that require a lot of data management or a lot of digital communication with other devices other than the XPS controller. In this case, it is more efficient to control the process from a dedicated program that runs on a PC and which sends (and receives) information to (and from) the XPS controller via the Ethernet TCP/IP communication interface. Communication to the XPS controller can be established from almost any PC and is independent of the PC's operating system (Windows, Linux, Unix, Mac OS, etc.) and programming language (LabVIEW, C++, C, Python, Matlab, VisualBasic, Delphi, etc.). The XPS controller supports the development of host-managed processes with a Windows communication DLL, a complete set of LabVIEW drivers and a number of example programs in C++, VisualBasic and LabVIEW. A few basic examples are provided in this section. For more details, please refer to the Software Drivers Manual.

XPS-managed processes (TCL)

The XPS controller is also capable of controlling processes directly using TCL scripts. TCL stands for Tool Command Language and is an open-sourced, object oriented, command language. With only a few fundamental constructs, it is very easy to learn and it is almost as powerful as C. Users of the XPS can use TCL to write a complete application code with any function. The TCL script can be executed in real time but in the background, utilizing time that the controller does not need for servo or communication. Multiple TCL programs run in a time sharing mode. To learn more about implementing TCL, refer to the TCL Manual which is accessible from the web site of the XPS controller.

The advantages of XPS-managed processes compared to host-managed processes are faster execution and better synchronization in many cases without any time taken from the communication link. XPS-managed processes or sub-processes are particularly valuable for repeating tasks, tasks that run in a continuous loop, and tasks that require a lot of data from the XPS controller. Examples include: anti-collision processes (processes that utilize security switches to stop motion when stages are in danger of collision); tracking, auto-focusing or alignment processes (processes that use external data inputs to control the motion); or custom initialization routines (processes that must constantly be executed during a system's use).

The XPS controller has real-time multi-tasking functionality, and with most applications there is not only a choice between a host-managed or an XPS-managed process, but also a recognition of splitting the application into the right number of sub-tasks, and defining the most efficient process for each sub-task. An efficient process design is one of the main challenges with today's most complex and critical applications in terms of time and precision. It is recommended to spend time thinking about the proper process definition and the best approach to control the XPS using a program.

However, not all details can be addressed in this chapter.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

18.1 TCL Generator

The TCL generator provides a convenient way of generating simple executable TCL scripts. These scripts are also a good place to start for the development of more complex scripts. Note that applications that are memory intensive or require links other XPS may require a script that is external to the XPS.

The TCL generator is accessible from the terminal page of the XPS web site. Pressing the TCL generator button generates a TCL script that includes the commands previously executed and listed in the Command history list. Note that the command order in the generated TCL script is the same, but in inverted order, as the executed commands in the Command history list. The name of the generated TCL script is History.tcl and is stored in the ..\Admin\Public\Scripts folder in the controller.

Be aware that generating a TCL script overwrites an already created History.tcl file if it is present on the XPS. To permanently save the created TCL script, connect to the XPS via FTP and either rename or move the History.tcl file before generating a new History.tcl from the 'TCL Generator" button.

Example

This is an example using three stages, two in an XY group (named XY) and one in a SingleAxis group (named S).

The following functions were executed in the Terminal web page.

KillAll()

GroupInitialize(S)

GroupInitialize(XY)

GroupHomeSearch(S)

GroupHomeSearch(XY)

GroupMoveAbsolute(S, 70)

GroupMoveAbsolute(S, -70)

GPIODigitalSet(GPIO3.DO, 63, 0)

EventExtendedConfigurationTriggerSet(XY.X.XYLineArcTrajectory.Start, 0, 0, 0, 0)

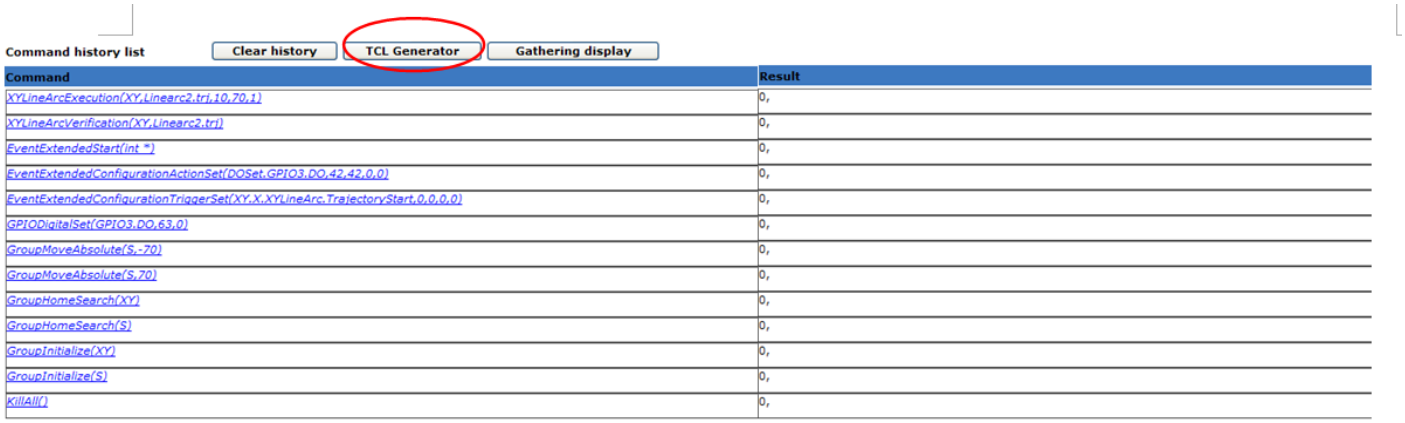
EventExtendedConfigurationActionSet(DOSet.GPIO3.DO, 42, 42, 0, 0)

EventExtendedStart()

XYLineArcVerification(XY, Linearc2.trj)

XYLineArcExecution(XY, Linearc2.trj, 10, 70, 1)

Then, the TCL Generator button is pressed to create a TCL script file. The file is named History.tcl. When executed, that TCL file will execute all of the functions used individually in the terminal.


Command history list | Clear history | TCL Generator | Gathering display

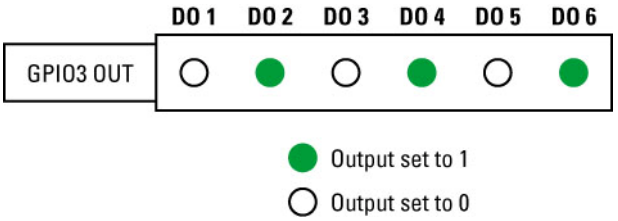
Command	Result
XYLineArcExecution(XY,Linearc2.trj,10,70,1)	0,
XYLineArcVerification(XY,Linearc2.trj)	0,
EventExtendedStart(int *)	0,
EventExtendedConfigurationActionSet(DOSet.GPIO3.DO,42,42,0,0)	0,
EventExtendedConfigurationTriggerSet(XY.X.XYLineArc.TrajectoryStart,0,0,0,0)	0,
GPIODigitalSet(GPIO3.DO,63,0)	0,
GroupMoveAbsolute(S,-70)	0,
GroupMoveAbsolute(S,70)	0,
GroupHomeSearch(XY)	0,
GroupHomeSearch(S)	0,
GroupInitialize(XY)	0,
GroupInitialize(S)	0,
KillAll()	0,

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To execute the script, use the XPS function TCLScriptExecute(History.tcl, task1, 0).

In this example, after initializing and homing both groups, the TCL script moves the single axis stage to the position of 70 units, then to the position of –70 units. It then sets all pins 1 - 6 on the digital output GPIO3 to 0.

Once checked, the line arc trajectory defined in the Linearc2.trj file gets executed with a velocity of 10 units/sec and an acceleration of 70 units/s^2. When this trajectory starts, more precisely when the positioner of the X axis starts moving, the bits #2, #4 and #6 of the output GPIO3 are set to 1 (42 = 101010).



NOTE

Selecting the function TCLScriptExecute() from the terminal menu opens a drop-down list for the available TCLFileNames. However, this list is limited to 100 entries.

To learn more about TCL programming, refer to the TCL Manual accessible from the documentation menu of the XPS web site. The TCL manual provides a complete description of all TCL commands and some more complex examples of TCL scripts.

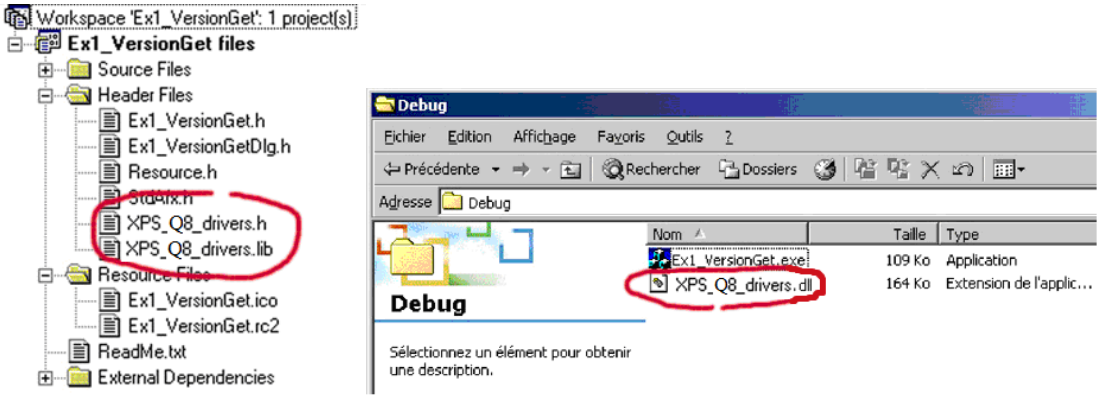
18.2 LabVIEW

LabVIEW is one of the most popular programming languages used with the XPS controller. Newport provides a complete set of LabVIEW drivers for the XPS controller. Refer to the XPS-Q8-LabVIEW Manual.pdf and the XPS-Q8-ProgrammerManual.pdf for additional details on implementing LabVIEW with the XPS and the location of the drivers.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

18.3 DLL Drivers

A DLL simplifies function calls from most programming languages. The DLL of the XPS controller is located in the ..Admin/Public/Drivers/DLL folder of the XPS controller. The files XPS_Q8_drivers.h and XPS_Q8_drivers.lib must be copied to the project folder and the file XPS_Q8_drivers.dll to the folder of the executable file.



Once these files are added, for instance to a C++ project, the prototypes of the functions can be called in the program with the respective syntax of the functions (parameters number, type…). The file XPS_Q8_drivers.h can be opened to see the list of the available functions and their prototypes.

For instance, the prototype of the function FirmwareVersionGet is as follows:

```
DLL int __stdcall FirmwareVersionGet (int SocketIndex, char * Version);
```

It requires two arguments (int and char*).

The maximum number of open sockets is 80. A supervisor can control as many XPS as much as the network and bandwidth can support.

Example C++ code

```
char  buffer [256] = {'\0'};
char pIPAddress[15] = {"192.168.33.236"};
int  nPort = 5001;
double dTimeOut = 60;
int  SocketID = -1;

// TCP / IP connection
SocketID = TCP_ConnectToServer(pIPAddress, nPort, dTimeOut);
if (SocketID == -1)
{
  sprintf (buffer, "Connection to @ %s, port = %ld failed\n", pIPAddress,
nPort);
  AfxMessageBox (buffer, MB_ICONSTOP);
}
else
{
  AfxMessageBox("Connected to target", MB_ICONINFORMATION);

  // Get controller version
FirmwareVersionGet (SocketID, buffer); // Get controller version
AfxMessageBox (buffer, MB_ICONINFORMATION);

  // TCP / IP disconnection
TCP_CloseSocket(SocketID);  // Close Socket
AfxMessageBox("Disconnected from target", MB_ICONINFORMATION);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This example opens a TCP connection, gets the firmware version of the XPS controller and closes the connection. The execution is displayed in message boxes:



To learn more about the DLL prototypes, refer to the *Programmer's Manual*, accessible from the web site interface of the XPS controller.

The *Software drivers manual*, also accessible from the XPS web site interface, provides further information about the use of the DLL and additional C++ programming examples.

18.4 Running Processes in Parallel

TCP provides a reliable, point-to-point communication channel that client-server applications on the Internet use to communicate with each other. To communicate over TCP, a client program and a server program establish a connection to one another. Each program binds a socket to its end of the connection. To communicate, the client and the server both read from and write to the socket that binds the connection.

Sockets are interfaces that can "plug into" each other over a network. Once "plugged in", the connected programs can communicate.

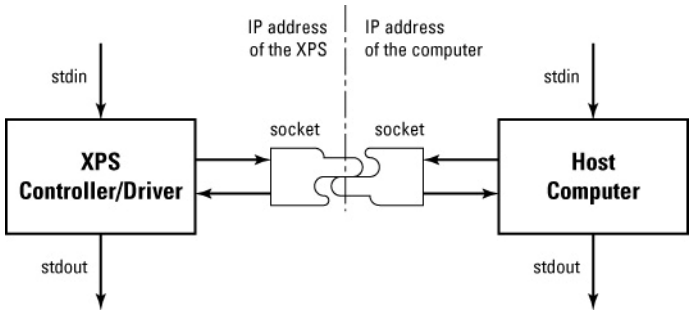


Figure 62: Running Processes in Parallel.

XPS uses blocking sockets. In other words, the programs/commands are "blocked" until the request for data has been satisfied. When the remote system writes data on the socket, the read operation will complete it and write the data in the received message window of the Terminal menu ('0' if command has been executed without error, or the error number in case of an error). That way, commands are executed sequentially since each command always waits for a response before finishing and then allowing execution of the next function. The main benefit of using this type of socket is that an execution acknowledgement is sent to the host computer with each function. In case of any error, it allows an exact diagnostic, which function has caused the error. It also allows a precise sequential process execution. On the other hand, more functions could be sent in parallel using non-blocking sockets. However, the drawback is that it is almost impossible to diagnose which function caused an error.

To execute several processes in parallel, for instance to request the current position during a motion and other data simultaneously, it is possible to communicate to the XPS controller via different sockets. The XPS controller supports a maximum number of 84 parallel opened sockets. The total number of open communication channels to the XPS controller, be it via the website, TCL scripts, a LabVIEW program, or any other program can not be larger than 84.

Users who prefer not to use blocking sockets, or whose programming languages don't support multiple sockets, such as Visual Basic versions prior to version .Net, can disable the blocking feature by setting a low TCPTimeOut value, 20 ms for instance. In

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

this case, the XPS will unblock the last socket after the TCPTimeOut time. However, this method loses the ability to pinpoint which commands were not properly executed.

Examples of the use of parallel sockets

The following examples illustrate how to open several sockets via the web site interface, TCL scripts, LabVIEW VIs and C++ programs.

Web site interface

The simplest way to open several sockets in parallel is to open several browser windows using the IP address of the controller. This is completely transparent to the user. Two or more groups of stages can be commanded from two terminal menus at the same time to execute different motions (multitasking).

TCL scripts

A TCL script is carried out sequentially: the commands are executed one by one following the order they are written in the script. Consequently, there is no benefit to open several sockets in a single TCL script.

However, it is possible to start a TCL script from another TCL script. That way, as many sockets and parallel processes can be started as needed. Below is an example with 3 open sockets:

```
######################
# TCL program : GEN #
######################

set TimeOut 10
set code 0
set Prog1 “ProgRV.tcl”
set Task1 “Task1”
set Prog2 “ProgXY.tcl”
set Task2 “Task2”

# open TCP socket
set code [catch “OpenConnection $TimeOut socketID”]

if {$code == 0} {
    puts stdout “ProgGen : TCP_ConnectToServer OK => $code   ID = $socketID”          <– Socket 1

set code [catch “TCLScriptExecute $socketID $Prog1 $Task1 0”]
puts stdout “ProgGen : TCLScriptExecute => error = $code”
                    <– Socket 2

set code [catch “TCLScriptExecute $socketID $Prog2 $Task2 0”]
puts stdout “ProgGen : TCLScriptExecute => error = $code”
                  <– Socket 3

# close TCP socket
set code [catch “TCP_CloseSocket $socketID”]
puts stdout “ProgGen : TCP_CloseSocket => $code  ID = $socketID”

} else  {
   puts stdout “ProgGen : TCP_ConnectToServer NOT OK => $code”
}
```

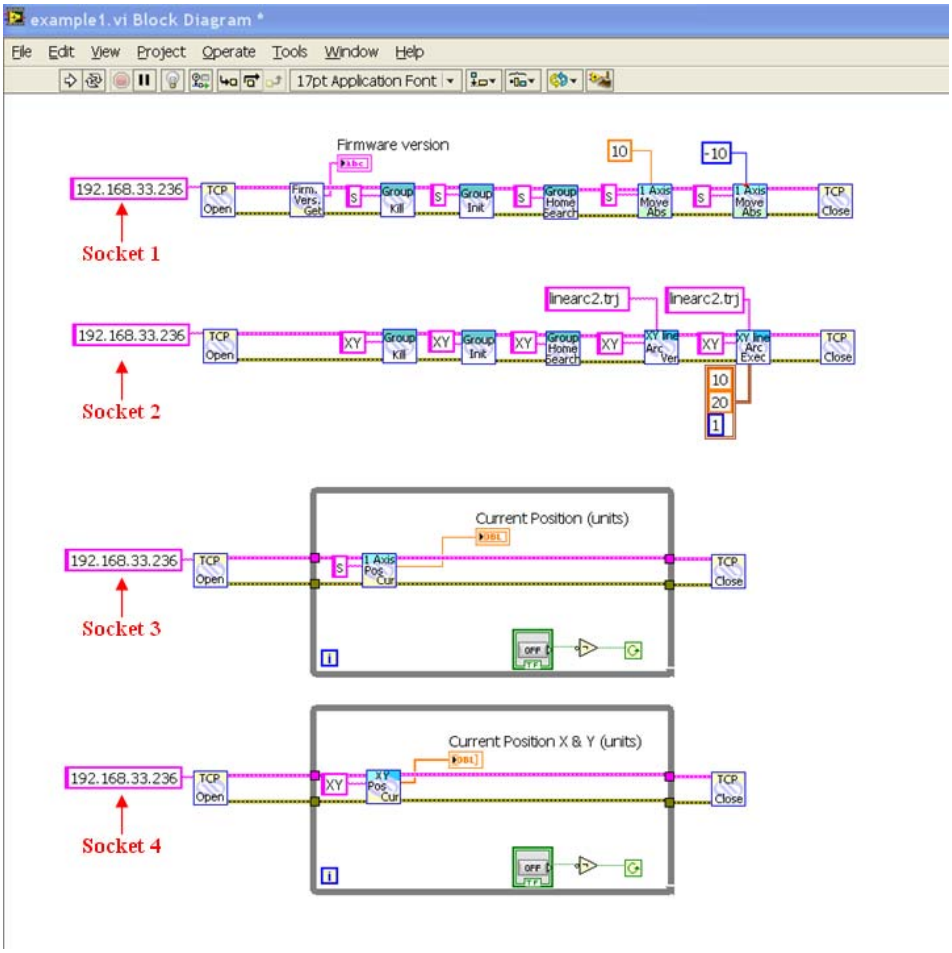

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE

Socket 2 and Socket 3 are not opened by the TCLScriptExecute function, but it is assumed that these scripts open a socket in their code.

LabVIEW VIs

In a VI file, several processes can easily be created, all beginning with a TCP Open and all finishing with a TCP Close. Each TCP Open will open its own socket. Shown below is a simple VI that opens 4 sockets simultaneously.



C++ program

A C++ program is executed sequentially. Even if it calls many functions, they are always executed one by one following the order they are written. In order to open several sockets for multitasking, the C++ multithreading functionality must be used.

The XPS driver DLL allows a maximum number of 100 simultaneously opened sockets. One XPS controller supports a maximum number of 84 simultaneously opened sockets, but a program could control several XPS controllers.

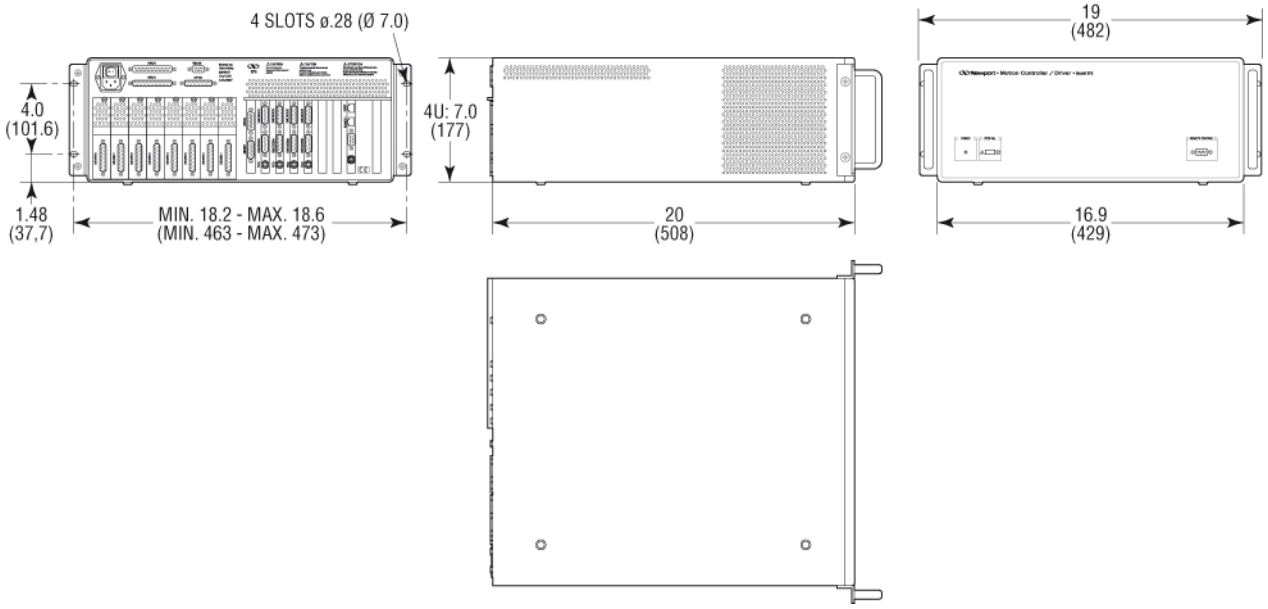
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Appendix

19.0 Appendix A: Hardware

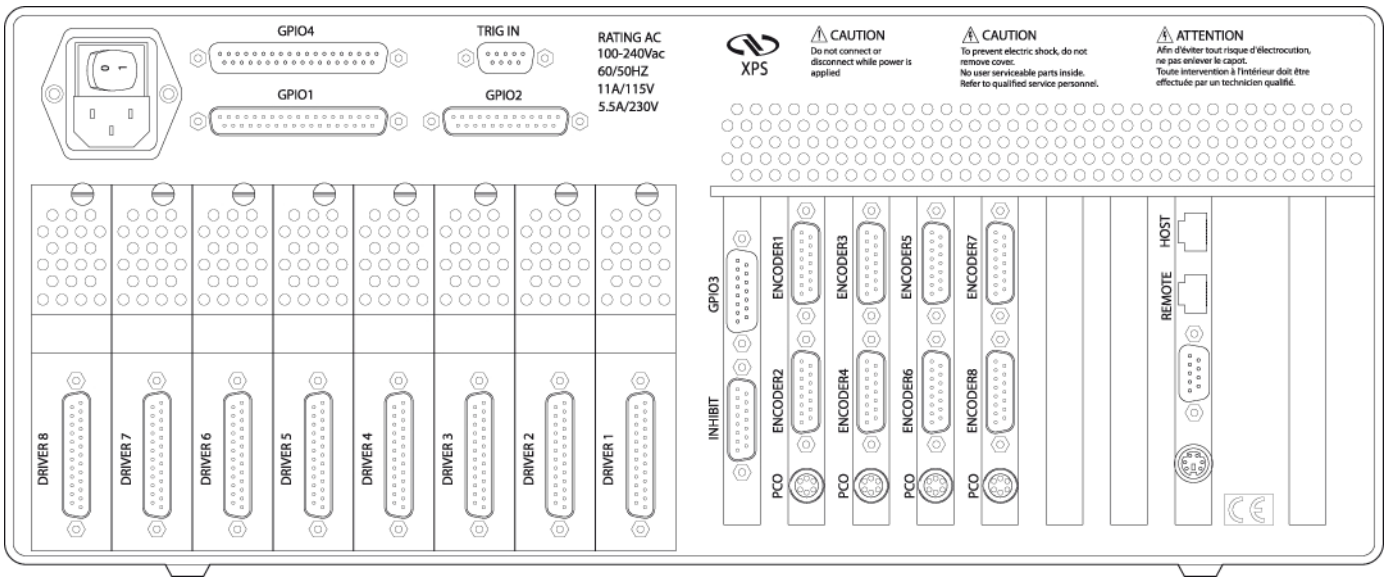
19.1 Controller



Weight:	16 kg (32 lb)
Input voltage:	100–240 VAC
Input current:	11 A/115 V 5.5 A/230 V
Frequency:	60/50 Hz

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

19.2 Rear Panel Connectors



19.3 Environmental Requirements

Temperature range:

Storage: -20 to +80 °C

Operating: +5 to +35 °C

Relative Humidity (Non-condensing):

Storage: 10 to 95% RH

Operating: 10 to 85% RH

Altitude:

Storage: To 10,000 ft

Operating: To 5,000 ft

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

20.0 Appendix B: General I/O Description

This paragraph briefly describes all XPS signal types.

A description of each XPS connector interface is detailed in further paragraphs.

20.1 Digital I/Os (All GPIO, Inhibit and Trigger In, and PCO Connectors)

All digital I/Os are TTL compatible:

- All digital I/Os are not isolated, but are referenced to electrical ground (GND).
- Input levels must be between 0 V and +5 V.
- Output levels should be at least +5 V (up to 30 V absolute maximum rating with open collector outputs).
- Outputs must be pulled up to the user's external power supply (+5 V to +24 V). This external power supply must be referenced to the XPS ground (GND).

All digital I/Os are refreshed asynchronously on user requests. Therefore, digital inputs or outputs have no refresh rate.

Typical delay is 100µs due to the clock cycle and priorities made to other functions.

All digital inputs are identical, except for GPIO3 inhibition input (described with GPIO3).

All digital inputs are in negative logic and have internal +5 V pull up resistors.

20.1.1 Digital Inputs

Parameter	Symbol	Min.	Max.	Units
Low Level Input Voltage	V_{IL}	0	0.8	V
High Level Input Voltage	V_{IH}	1.6	5	V
Input Current LOW	I_{IL}	–	-2.5	mA
Input Current HIGH	I_{IH}	–	0.4	mA

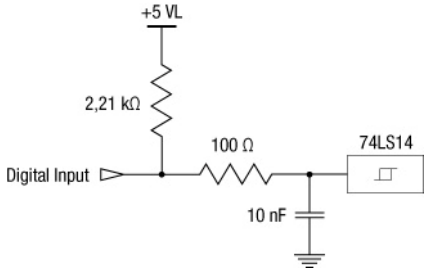


Figure 63: Digital TTL Input.

GPIOn inputs (n = 1 to 4) can be accessed via the GPIODigitalGet(GPIOn.DI, …) function.

All digital outputs are identical.

All digital outputs are in negative logic (NPN open collector, 74LS06 TTL type circuit) and have no internal pull up to permit levels above +5 V.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

20.1.2 Digital Outputs

Parameter	Symbol	Min.	Max.	Units
Low Level Output Voltage	V_{OL}	0	1	V
High Level Output Voltage	V_{OH}	2.4	30	V
Input Current LOW	I_{OL}	–	-40	mA
Input Current HIGH	I_{OH}	–	0.2	mA

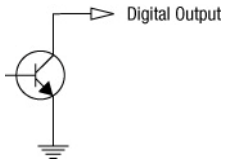


Figure 64: Open Collector Digital Output.

GPIOn outputs (n = 1 to 4) can be accessed via the GPIODigitalSet(GPIOn.DO, …) function.

20.2 Digital Encoder Inputs (Driver Boards & DRV00)

All digital encoder inputs are RS-422 standard compliant:

- All digital encoder signals are not isolated, but are referenced to the electrical ground (GND).
- Encoder signals must be differential pairs (using 26LS31 or MC3487 line driver type circuits). Encoder inputs have a terminating impedance of 120 Ω.
- Inputs are always routed on differential pairs. For a high level of signal integrity, we recommend using shielded twisted pairs of wires for each differential signal.
- Encoder power supply is +5 V @ 250 mA maximum (referenced to the electrical ground) and is sourced directly by the driver boards.

20.3 Digital Servitudes (Driver Boards, DRV00 & Analog Encoders Connectors)

All servitude inputs are TTL compatible:

- All servitude inputs are not isolated, but are referenced to the electrical ground (GND).
- Input levels must be between 0 V and +5 V.

All servitude inputs are refreshed synchronously with the XPS control loop (10 kHz).

All servitude inputs are identical.

All servitude inputs expect normally closed sensors referenced to ground (input is activated if the sensor is open) and have internal 2.2 kΩ pull up resistors to the +5 V.

20.4 Analog Encoder Inputs (Analog Encoder Connectors)

The analog encoder interface complies with the Heidenhain LIF481 glass scales wiring standard.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

20.5 Analog I/O (GPIO2 Connector)

20.5.1 Analog Inputs

The 4 analog inputs have a range of ±10 V, 14 Bit resolution, and a 15 kHz 2nd order low pass filter.

In all cases, the analog input values must be within the ±10 V range. The analog input impedance is typically 22 kΩ. The maximum input current is ±500 µA.

1 LSB = 20 V/16384 ≈ 1.22 mV

The maximum offset error is ±17.1 mV.

20.5.2 Analog Outputs

The 4 analog outputs have a range of ±10 V and 16 Bit resolution. The maximum offset error is ±2 mV, and the maximum gain error is ±6 LSB. The output settling time is typically 50 µsec at 1% of the target value (output filter is a 15 kHz 1st order low pass filter).

Analog outputs are voltage outputs (output current less than 1 mA), so to use them properly, they must be connected to an impedance higher than 10 kΩ.

1 LSB = 20 V/65536 ≈ 0.3 mV.

Analog outputs can be accessed via the GPIOAnalogSet(GPIO2.DACn,…) function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

21.0 Appendix C: Power Inhibit Connector



PIN	Signal type	Description	PIN	Signal type	Description
1	Supply	GND	9	Reserved	
2	Inhibition	Must be connected to GND	10	Reserved	
3	Reserved		11	Reserved	
4	Reserved		12	Reserved	
5	Reserved		13	Reserved	
6	Reserved		14	Reserved	
7	Reserved		15	Reserved	
8	Reserved				

Figure 65: Inhibition connector.

This connector is provided for the wiring of a remote STOP ALL switch.

It has the same effect as the front panel STOP ALL button.

Inhibition input is a standard TTL input.

Inhibition (Pin #2), must always be connected to GND during normal controller operation.

An open circuit is equivalent to pressing STOP ALL on the front panel. Wire the switch contacts normally closed.

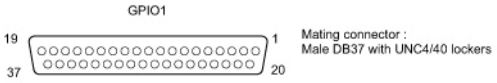
NOTE

Connecting more than one switch is not recommended on this input.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

22.0 Appendix D: GPIO Connectors

22.1 GPIO1 Connector



PIN	Signal type	Description	PIN	Signal type	Function
1	N.C.		20	Supply	GND
2	Supply	+12V supply output<100mA	21	Supply	GND
3	Supply	+5V supply output<100mA	22	Supply	GND
4	TTL input	Input 1	23	Supply	GND
5	TTL input	Input 2	24	Supply	GND
6	TTL input	Input 3	25	Supply	GND
7	TTL input	Input 4	26	Supply	GND
8	TTL input	Input 5	27	Supply	GND
9	TTL input	Input 6	28	Supply	GND
10	TTL input	Input 7	29	Supply	GND
11	TTL input	Input 8	30	Supply	GND
12	O.C. output	Output 1	31	Supply	GND
13	O.C. output	Output 2	32	Supply	GND
14	O.C. output	Output 3	33	Supply	GND
15	O.C. output	Output 4	34	Supply	GND
16	O.C. output	Output 5	35	Supply	GND
17	O.C. output	Output 6	36	Supply	GND
18	O.C. output	Output 7	37	Supply	GND
19	O.C. output	Output 8			

Figure 66: GPIO1 Digital I/O Connector.

General Purpose Inputs Outputs GPIO1 is the main XPS digital I/O connector.

22.2 GPIO2 Connector



PIN	Signal type	Description	PIN	Signal type	Description
1	Supply	GND	14	Analog input	Analog input 1
2	N.C.		15	Analog input	Analog input 2
3	TTL input	Input 1	16	Analog input	Analog input 3
4	TTL input	Input 2	17	Analog input	Analog input 4
5	TTL input	Input 3	18	Supply	GND
6	TTL input	Input 4	19	Analog output	Analog output 1
7	TTL input	Input 5	20	Analog output	Analog output 2
8	N.C.		21	Analog output	Analog output 3
9	N.C.		22	Analog output	Analog output 4
10	Reserved		23	Supply	GND
11	Reserved		24	TTL input	Input 6
12	Reserved		25	Supply	GND
13	Supply	GND			

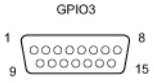
Figure 67: GPIO2 Analog & Digital Connector.

General Purpose Inputs Outputs GPIO2 is an additional digital input connector.

This connector is also the main analog I/O connector with 4 analog inputs and 4 analog outputs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

22.3 GPIO3 Connector

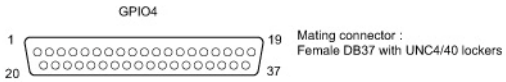


PIN	Signal type	Function	PIN	Signal type	Function
1	Supply	GND	9	TTL input	Input 1
2	Reserved		10	TTL input	Input 2
3	O.C. output	Output 1	11	TTL input	Input 3
4	O.C. output	Output 2	12	TTL input	Input 4
5	O.C. output	Output 3	13	TTL input	Input 5
6	O.C. output	Output 4	14	TTL input	Input 6
7	O.C. output	Output 5	15	Supply	+5V supply output<100mA
8	O.C. output	Output 6			

Figure 68: GPIO3 Digital I/O Connector.

General Purpose Inputs Outputs GPIO3 is a digital I/O connector.

22.4 GPIO4 Connector



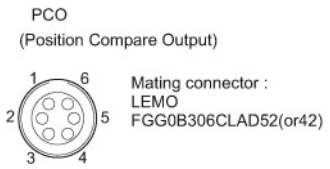
PIN	Signal type	Description	PIN	Signal type	Description
1	TTL input	Input 1	20	TTL input	Input 2
2	TTL input	Input 3	21	TTL input	Input 4
3	TTL input	Input 5	22	TTL input	Input 6
4	TTL input	Input 7	23	TTL input	Input 8
5	TTL input	Input 9	24	TTL input	Input 10
6	TTL input	Input 11	25	TTL input	Input 12
7	Supply	GND	26	Supply	+5V supply output <100mA
8	Supply	GND	27	O.C. output	Output 1
9	O.C. output	Output 2	28	O.C. output	Output 3
10	O.C. output	Output 4	29	O.C. output	Output 5
11	O.C. output	Output 6	30	Supply	+5V supply output <100mA
12	Supply	GND	31	TTL input	Input 13
13	TTL input	Input 14	32	TTL input	Input 15
14	TTL input	Input 16	33	O.C. output	Output 7
15	O.C. output	Output 8	34	O.C. output	Output 9
16	O.C. output	Output 10	35	O.C. output	Output 11
17	O.C. output	Output 12	36	O.C. output	Output 13
18	O.C. output	Output 14	37	O.C. output	Output 15
19	O.C. output	Output 16			

Figure 69: GPIO4 Additional Digital I/O Connector.

General Purpose Inputs Outputs GPIO4 is an additional digital I/O connector.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

23.0 Appendix E: PCO Connector



PIN	Distance spaced pulses or Tim spaced pulses	AquadB signals
1	+5 VL	+5 VL
2	Axis 1: Pulse enable	Axis 1: A
3	Axis 1: Pulse	Axis 1: B
4	Axis 2: Pulse Enable	Axis 2: A
5	Axis 2: Pulse	Axis 2: B
6	GND	GND

Figure 70: Position Compare Output Connector.

There is one PCO connector for every two axes. Axis #1 refers to the upper (odd) encoder plug and axis #2 refers to the lower (even) encoder plug. The signals provided on this plug depend on the configuration of the output triggers, see chapter 13, Output trigger, for more details.

The state of the enable signal is low when the stage is inside the programmed position compare window.

Note also, that only the falling edge of the trigger pulse is precise and only this edge should be used for synchronization regardless of the PCOPulseWidth setting.

The duration of the pulse is 200 nsec by default and can be modified using the function **PositionerPositionComparePulseParametersSet()**. Possible values for the PCOPulseWidth are: 0.2 (default), 1, 2.5 and 10 (µs). Successive trigger pulses should have a minimum time lag equivalent to the PCOPulseWidth time times two.

The signals are open collector type and accept up to 30 Volts and 40 mA

The +5V output provided on the PCO connector can be used to pull-up these outputs and can supply 50 mA max.

NOTE

To ensure fast transitions with an open collector, it is necessary to have enough current to speed-up the transistor's junction capacitor charge / discharge. A good value is around 10 mA. So to pull-up the PCO signals to +5 V, a 470 Ω resistor can be used.

Refer to section B.1 Digital I/Os, Digital Outputs for detailed electrical description.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

24.0 Appendix F: Motor Driver Cards

24.1 DC and Stepper Motor Driver XPS-DRV01



PIN	Function		PIN	Function	
	D.C. MOTOR	STEPPER MOTOR		D.C. MOTOR	STEPPER MOTOR
1	Tachometer+	Ph 1	15	Index	Index
2	Tachometer+	Ph 1	16	Limit ground	Limit ground
3	Tachometer-	Ph 2	17	+Travel limit	+Travel limit
4	Tachometer-	Ph 2	18	-Travel limit	-Travel limit
5	Motor+	Ph 3	19	Encoder A	Encoder A
6	Motor+	Ph 3	20	Encoder B	Encoder B
7	Motor-	Ph 4	21	+5V	+5V
8	Motor-	Ph 4	22	Encoder ground	Encoder ground
9	Reserved	Common 3&4	23	Encoder /A	Encoder /A
10	Reserved	Common 3&4	24	Encoder /B	Encoder /B
11	Reserved	Common 1&2	25	/index	/index
12	Reserved	Common 1&2			
13	Origin	Origin			
14	Shld GND	Shld GND			

Figure 71:XPS-DRV01 Motor Driver Connectors.

Motor + This output must be connected to the positive lead of the DC motor. The voltage seen at this pin is pulse-width modulated with maximum amplitude of 48 V DC.

Motor - This output must be connected to the negative lead of the DC motor. The voltage seen at this pin is pulse-width modulated with maximum amplitude of 48 V DC.

Ph1 This output must be connected to Winding A+ lead of a two-phase stepper motor. The voltage seen at this pin is pulse-width modulated with maximum amplitude of 48 V DC.

Ph2 This output must be connected to Winding A- lead of a two-phase stepper motor. The voltage seen at this pin is pulse-width modulated with maximum amplitude of 48 V DC.

Ph3 This output must be connected to Winding B+ lead of a two-phase stepper motor. The voltage seen at this pin is pulse-width modulated with maximum amplitude of 48 V DC.

Ph4 This output must be connected to Winding B- lead of a two-phase stepper motor. The voltage seen at this pin is pulse-width modulated with maximum amplitude of 48 V DC.

Common 3&4 This output must be connected to the center tab of Winding B of a two-phase stepper motor. The voltage seen at this pin is pulse-width modulated with maximum amplitude of 48 V DC.

Common 1&2 This output must be connected to the center tab of Winding A of a two-phase stepper motor. The voltage seen at this pin is pulse-width modulated with maximum amplitude of 48 V DC.

+ Travel limit This input is pulled-up to +5 V with a 2.2 kΩ resistor by the controller and represents the stage positive direction hardware travel limit.

- Travel limit This input is pulled-up to +5 V with a 2.2 kΩ resistor by the controller and represents the stage negative direction hardware travel limit.

Encoder A & /A These A and /A inputs are differential inputs. Signals are compliant with RS422 electrical standard and are received with a

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

26LS32 differential line receiver. A resistor of 120 Ω adapts the input impedance. The A and /A encoder signals originate from the stage position feedback circuitry and are used for position tracking.

Encoder B and /B These B and /B inputs are differential inputs. Signals are compliant with RS-422 electrical standard and are received with a 26LS32 differential line receiver. A resistor of 120 Ω adapts the input impedance. The B and /B encoder signals originate from the stage position feedback circuitry and are used for position tracking.

Index & /Index These Index and /Index inputs are differential inputs. Signals are compliant with RS422 electrical standard and are received with a 26LS32 differential line receiver. A resistor of 120 Ω adapts the input impedance. The Index and /Index signals originate from the stage and are used for homing the stage to a repeatable location.

Encoder ground Ground reference for encoder feedback.

Origin This input is pulled-up to +5 V with a 2.2 kΩ resistor by the controller. The Origin signal originates from the stage and is used for homing the stage to a repeatable location.

+5 V (DRV01: 250 mA Maximum) A +5 V DC supply is available from the driver. This supply is provided for stage home, index, travel limit, and encoder feedback circuitry.

Limit ground Ground for stage travel limit signals. Limit ground is combined with digital ground at the controller side.

Shield GND Motor cable shield ground.

Brake + (available only on DRVM board) Voltage command (24 V or 48 V: strap on the driver board) to drive the brake.

Brake – (available only on DRVM board) Reference of the above voltage command.

Tachometer + & Tachometer – These inputs are used to receive tachometer voltage information. This voltage depends on the output voltage rating of the employed tachometer.

24.2 Three phase AC brushless driver XPS-DRV02

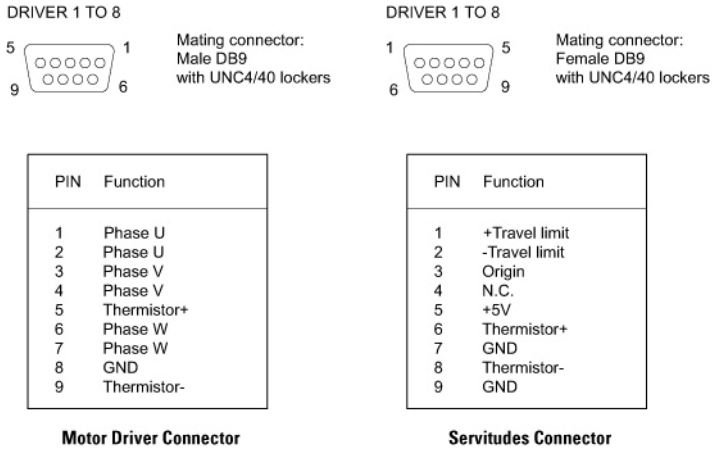
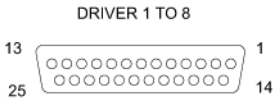


Figure 72: XPS-DRV02 Motor Driver Connectors.
The stage thermistor can be connected to either connector.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

24.3 DC Motor Driver XPS-DRV03



PIN	Function	PIN	Function
1	Tachometer+	15	Index
2	Tachometer+	16	Limit ground
3	Tachometer-	17	+Travel limit
4	Tachometer-	18	-Travel limit
5	Motor+	19	Encoder A
6	Motor+	20	Encoder B
7	Motor-	21	+5V
8	Motor-	22	Encoder ground
9	N.C.	23	Encoder /A
10	N.C.	24	Encoder /B
11	N.C.	25	/index
12	N.C.		
13	Origin		
14	Shld GND		

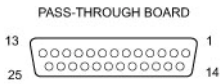
Figure 73: XPS-DRV03 Motor Driver Connectors.

24.4 Pass-Through Board Connector (25-Pin D-Sub) XPS-DRV00



WARNING

The Pass-through board connector replaces the motor interface connector only if the axis is connected to an external motor driver.



PIN	Function	PIN	Function
1	Reserved	14	Reserved
2	+5 V	15	Inhibition output
3	Origin input	16	Reserved
4	– Travel limit input	17	Reserved
5	+ Travel limit input	18	Reserved
6	Main fault input	19	Encoder /A input
7	Encoder A input	20	Encoder /B input
8	Encoder B input	21	/Index input
9	Index input	22	Reserved
10	Pulse / Pulse+ output	23	GND
11	Direction / Pulse- output	24	N.C.
12	Analog A output	25	GND
13	Analog B output		

Figure 74: DRV00 Pass-Through Connector.

Analog A output and Analog B output have 16 bit resolution at ±10 V output. These signals are used to command an external driver.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

25.0 Appendix G: Analog Encoder Connector



PIN	Function	PIN	Function
1	Analog VA (0.5Vpp)	9	Analog /VA (0.5Vpp)
2	0V	10	0V
3	Analog VB (0.5Vpp)	11	Analog /VB (0.5Vpp)
4	+5VA	12	+5VL
5	N.C.	13	N.C.
6	Limit (TTL input)	14	Analog VI (0.5Vpp)
7	Analog /VI (0.5Vpp)	15	N.C.
8	Home (TTL input)		

Figure 75: Analog Encoders Connector.

This connector is used to receive sine/cosine encoder signals.

The sinusoidal position signals, sine and cosine, must be phase-shifted by 90° and have signal levels of approximately 1 Vpp. Each of these two signals is composed of an analog sinusoidal signal and its complement entering in a differential amplifier (Sine = Analog VA - Analog /VA).

Analog VA, Analog /VA, Analog VB, Analog /VB, Analog VI and Analog /VI:

Levels for these individual signals must be 0.5 Vpp.

VA, /VA, VB and /VB inputs are the sine and cosine signals from the encoder glass scale.

VI and /VI inputs are used to receive Index information from the encoder glass scale.

+5 VA:

This +5 V DC supply is provided for supplying the encoder.

+5 VL:

This +5 V DC supply is provided for supplying digital circuits (Limit and Home).

Limit and Home are TTL inputs for Limit switch management and homing purposes directly from the encoder glass scale.

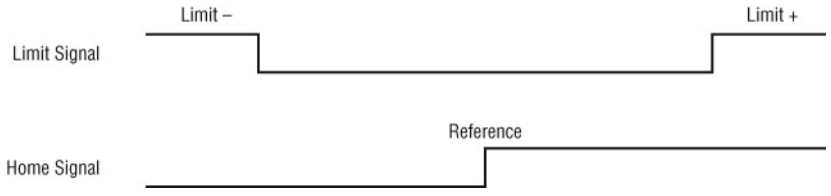


Figure 76: Limit and Home TTL Input Signals.

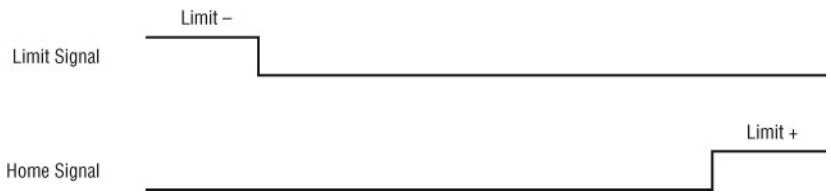
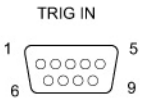


Figure 77: Limit and Limit TTL Input Signals.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

26.0 Appendix H: Trigger IN Connector



Mating connector :
Female DB9 with UNC4/40 lockers

PIN	Function	PIN	Function
1	GND	6	Syncro input
2	GND	7	N.C.
3	N.C.	8	N.C.
4	N.C.	9	N.C.
5	N.C.		

Figure 78: Trigger Input Connector.

Synchro is a TTL input. It is used to trigger acquisition of the XPS controller acquisition (External gathering).

A low to high transition will latch all encoders and analog inputs in the controller.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Service Form

Your Local Representative

Tel.: _____

Fax: _____

Name: _____

Company: _____

Address: _____

Country: _____

P.O. Number: _____

Item(s) Being Returned: _____

Model#: _____

Return authorization #: _____
(Please obtain prior to return of item)

Date: _____

Phone Number: _____

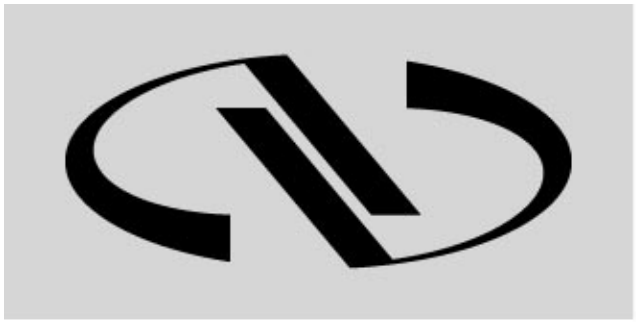
Fax Number: _____

Serial #: _____

Description: _____

Reasons of return of goods (please list any specific problems): _____

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Newport®

Experience | Solutions

Visit Newport Online at:
www.newport.com

North America & Asia
Newport Corporation
1791 Deere Ave.
Irvine, CA 92606, USA

Sales
Tel.: (800) 222-6440
e-mail: sales@newport.com

Technical Support
Tel.: (800) 222-6440
e-mail: tech@newport.com

Service, RMAs & Returns
Tel.: (800) 222-6440
e-mail: rma.service@newport.com

Europe
MICRO-CONTROLE Spectra-Physics S.A.S
9, rue du Bois Sauvage
91055 Évry CEDEX
France

Sales
Tel.: +33 (0)1.60.91.68.68
e-mail: france@newport-fr.com

Technical Support
e-mail: tech_europe@newport.com

Service & Returns
Tel.: +33 (0)2.38.40.51.55

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP