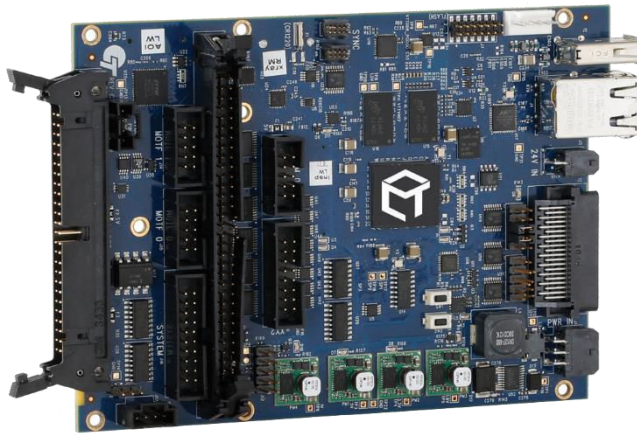


ScanMaster Controller

Advanced Laser Positioning and Control for Laser Steering Systems

Software Reference Manual



Read carefully before using.
Retain for future reference.

TABLE OF CONTENTS

Table of Contents	2
List of Tables	8
1 Important Information	1
1.1 Safety Symbols	1
1.2 Safety Labels	2
1.3 General Safety Guidelines	2
1.4 Customer Support	3

2 Introduction	5
2.1 General Notes	5
2.2 Using This Manual	5
2.2.1 Purpose	5
2.2.2 Revision History	6
2.3 Warranty Information	9

3 SMC Product introduction	11
3.1 System Description	11
3.2 Feature Overview	12
3.2.1 Hardware Features	12
3.2.2 Software Features	13
3.3 Application Programming Interface	13
3.3.1 Installation Location	14
3.3.2 API Structure	15
3.3.3 Win32 C++ Interfaces	16

4 Software Overview	18
4.1 The Use of XML in the API	20

5 Broadcast API	22
-----------------	----

Table of Contents

5.1	Establishing a Connection	22
5.1.1	clientAttachBroadcast	22
5.1.2	clientDetachBroadcast	23
5.2	Retrieving Broadcast Data	24
5.2.1	getServerCount	24
5.2.2	getServerList	25
5.2.3	getBroadcastData	26
5.3	Broadcast Data Definitions	27
5.3.1	Broadcasted System Information	28
5.3.2	Broadcasted Status Information	32
<hr/>		
6	Session API	36
6.1	Access to SMC Modules	36
6.1.1	loginSession	36
6.1.2	logoutSession	37
6.2	Configuration Data Management	38
6.2.1	getFixedDataList	39
6.2.2	requestFixedData	41
6.2.3	sendFixedData	42
6.3	Configuration Data Definitions	43
6.3.1	Administration Configuration	45
6.3.2	Controller Configuration	51
6.3.3	Laser Configuration	58
6.3.4	Lens Configuration	67
6.3.5	Correction Tables	70
6.3.6	User Configuration	83
6.3.7	Performance Adjustments	85
6.3.8	Servo Configuration	86
6.4	Marking Job Specification	89
6.4.1	Job Data Types	89
6.4.2	Job Data Definition	89
6.4.3	Job Type Specification	91
6.5	Job Parameters and Commands	91
6.5.1	User Units Conversion	91
6.5.2	Motion Control Parameters	94

Table of Contents

6.5.3	Motion Control Commands	109
6.5.4	Laser Control Parameters	126
6.5.5	Laser Control Commands	132
6.5.6	External I/O Commands	135
6.5.7	Utility Commands	138
6.5.8	Coordinate System Transform Parameters	142
6.5.9	Hardware Interface Configuration Parameters	148
6.5.10	Bit-map Raster Support	155
6.5.11	Bit-map Raster Commands	161
6.5.12	Polygon Bit-map Raster Commands	164
6.5.13	Mark-on-the-fly Support	169
6.5.14	Velocity Controlled Laser Modulation	183
6.5.15	Via-hole Drilling Support	189
6.6	Structured Job Organization	203
6.6.1	Segment Construct	204
6.6.2	Structured Job Sequencing	205
6.6.3	Structured Job Example	208
6.7	Marking Job Control and Administration	212
6.7.1	sendStreamData (overload 1)	212
6.7.2	sendStreamData (overload 2)	213
6.7.3	sendCorrectionData (overload 1)	215
6.7.4	sendCorrectionData (overload 2)	216
6.7.5	sendCorrectionData (overload 3)	217
6.7.6	saveJobData	219
6.7.7	sendJobData	220
6.7.8	copyJobData	221
6.7.9	manageJobData	221
6.7.10	requestJobNameList	222
6.8	Asynchronous Communication	223
6.8.1	OnConnectEvent	224
6.8.2	OnMessageEvent	224
6.8.3	OnDataEvent	231
6.9	Priority Communication	232
6.9.1	sendPriorityData	232
6.9.2	Priority Messages	233
6.9.3	getPriorityData	242

Table of Contents

6.9.4	GetRegisters Priority Message OnDataEvent Response	243
6.9.5	GetCalFactors Priority Message OnDataEvent Response	245
6.10	API Error Codes	245
<hr/>		
7	Remote Control API	247
7.1	TCP/IP Interface	247
7.2	RS232 Interface	248
7.3	Protocol Specification	248
7.3.1	Control and Communications Commands	249
7.3.2	Job Execution Control	263
7.3.3	System Administration Commands	270
7.4	Remote Control Return Codes	279
<hr/>		
8	Appendix A - Theory of Operation	280
8.1	Scanning Job Fundamentals	280
8.1.1	Coordinate System Conventions	280
8.1.2	Marks and Jumps	281
8.1.3	Laser Marking Terms and Definitions	282
8.1.4	Micro-Vectoring	283
8.1.5	Delays	283
8.2	Image Field Correction	289
8.2.1	X-Y Mirror Induced Distortion	290
8.2.2	F-theta Objective Induced Distortion	291
8.2.3	Composite Distortion and Correction Methodology	292
8.2.4	Multiple Correction Table Support	292
8.3	Laser Timing Control	293
8.4	Software Control of Laser Timing	296
8.4.1	Laser Timing Emulation	299
<hr/>		
9	Appendix B - Error Codes	315
9.1	XML API Error Codes	315
9.2	Remote Control Error Codes	317
9.3	LastError Code Descriptions	319
<hr/>		
10	Index	323

LIST OF FIGURES

Figure 1 - Client-Server Architecture.....	19
Figure 2 - SMC Software Data Flow.....	19
Figure 3 - SMC Configuration File Relationships	39
Figure 4 - “Fire-on-the-fly”, Mode 0.....	156
Figure 5 - “Fire-on-the-fly”, Mode 1.....	157
Figure 6 - Standard “Jump-and-fire” Mode.....	159
Figure 7 - Synchronous “Jump-and-fire” Mode.....	161
Figure 8 - Mark-on-the-fly Basic Process Flow	178
Figure 9 - Mark-on-the-fly Usage in Wire Marking	179
Figure 10 - Mark-on-the-fly Usage in Multi-image-field Applications.....	181
Figure 11 - Velocity Controlled Laser Modulation Overview	184
Figure 12 - Velocity Controlled Laser Modulation: Duty-cycle, Acceleration Effect	185
Figure 13 - Velocity Controlled Laser Modulation: Duty-cycle, Deceleration Effect.....	185
Figure 14 - Velocity Controlled Laser Modulation: Frequency, Acceleration Effect	186
Figure 15 - Velocity Controlled Laser Modulation: Frequency, Deceleration Effect.....	187
Figure 16 - Velocity Controlled Laser Modulation: Laser Power	187
Figure 17 - Interlock Sequencing.....	230
Figure 18 - Scanning System Coordinate Conventions.....	280
Figure 19 - Laser Marking Sample	281
Figure 20 - Micro-vector Operation	283
Figure 21 - Micro-vectoring and Laser Timing Relationships	289
Figure 22 - Projection System Layout.....	290
Figure 23 - Pincushion Distortion Caused by an X-Y Mirror Set	291
Figure 24 - Pillow Distortion Caused by F-theta Lens.....	291
Figure 25 - Composite Image Field Distortion.....	292

Table of Contents

Figure 26 - Multiple Correction Table Usage in the SMC	293
Figure 27 - Laser Timing Relationships	294
Figure 28 - Laser Timing for CO ₂ Laser Systems	300
Figure 29 - Nd:YAG Emulation Mode-1 (Raylase Nd:YAG Mode-1 and Scanlab YAG 1).....	302
Figure 30 - Nd:YAG Emulation Mode-2 (Raylase Nd:YAG Mode-2)	304
Figure 31 - Nd:YAG Emulation Mode-3 (Raylase Nd:YAG Mode-3)	306
Figure 32 - Nd:YAG Emulation Mode-4 (Scanlab YAG 2).....	308
Figure 33 - Nd:YAG Emulation Mode-5 (Scanlab YAG 3).....	310
Figure 34 - Fiber Laser Timing	312

LIST OF TABLES

Table 1 - Revision History	6
Table 2 - SMC API DLLs	14
Table 3 - Sample XML Statements.....	20
Table 4 - Broadcast Data Types	27
Table 5 - Data Type Keys	28
Table 6 - Broadcasted System Information	28
Table 7 - State Code Descriptions	31
Table 8 - Broadcasted Status Information.....	32
Table 9 - Fixed Data Type Codes	43
Table 10 - Administration Configuration Data	45
Table 11 - Controller Configuration Data	52
Table 12 - Laser Configuration Data: Header and Host Application Initialization Settings.....	59
Table 13 - Hardware Initialization Settings	61
Table 14 - Lens Configuration Data: Header and Host Application Initialization Settings	67
Table 15 - Lens Configuration Data: Hardware Initialization Settings	69
Table 16 - Correction Table Parametric Information	71
Table 17 - Correction Table Hardware Initialization Settings.....	81
Table 18 - User Configuration Data Settings: Header and Host Application Initialization	83
Table 19 - User Configuration Data: Hardware Initialization Settings	84
Table 20 - Performance Adjustments Data Header	85
Table 21 - Performance Adjustments Data: Hardware Initialization Settings.....	86
Table 22 - Servo Config Data	87
Table 23 - Structured Job Example.....	208
Table 24 - OnMessageEvent Message Types	225
Table 25 - Predefined Application Message Event Codes	226

List of Tables

Table 26 - Priority Message Descriptions.....	233
Table 27 - Laser Marking Terms and Definitions.....	282
Table 28 - Delay Parameters	284
Table 29 - Laser Configuration Control XML Examples	296
Table 30 - Example CO ₂ Laser Configuration XML.....	301
Table 31 - Example Nd:YAG Mode-1 Laser Configuration XML.....	303
Table 32 - Example Nd:YAG Emulation Mode-2 Laser Configuration XML	305
Table 33 - Example Nd:YAG Mode-3 Laser Configuration XML.....	306
Table 34 - Example Nd:YAG Mode-4 Laser Configuration XML.....	308
Table 35 - Example Nd:YAG Mode-5 Laser Configuration XML.....	310
Table 36 - Example IPG Fiber Laser Configuration XML	313
Table 37 - API Error Codes.....	315
Table 38 -	315
Table 39 - Remote Control Return Codes.....	317
Table 40 - LastError Code Descriptions	319

1 IMPORTANT INFORMATION



For your protection, carefully read these instructions before installing and operating the scan head.

Retain these instructions for future reference.

Novant reserves the right to update this user manual at any time without prior notification.

If product ownership changes, this manual should accompany the product.

1.1 SAFETY SYMBOLS

This manual uses the following symbols and signal words for information of importance.



DANGER

Indicates a hazardous situation which, if not avoided, will result in serious injury or death.



WARNING

Indicates a hazardous situation which, if not avoided, could result in serious injury or death.



CAUTION

Indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.



IMPORTANT

Indicates information considered important but not directly hazard related (e.g. security, hygiene, or equipment or property damage).

1.2 SAFETY LABELS



DANGER

Laser radiation

can cause severe retinal and corneal burns, burns on the skin, and may pose a fire risk.

- To avoid injury and reduce risk of fire, please follow the control measures and safety guidelines provided by the laser's manufacturer, and those established by your Laser Safety Officer (LSO), Radiation Safety Officer (RSO), or safety department of your business or institution.



ESD WARNING

Electrostatic discharge and improper handling
can damage MOVIA scan head's electronics.

- Keep the equipment sealed until it is located at a proper static control station.

1.3 GENERAL SAFETY GUIDELINES



Laser Radiation

Do not stare directly into a laser beam.

Follow all system laser safety requirements during installation and operation.

Shutter Safety



Where practical, Novanta recommends the use of an internal shutter mechanism to prevent unwarranted emission of laser radiation. If this is not possible, consult the laser vendor to design a proper safety shutter that, when activated, will eliminate all possibility of exposure exceeding Class 1 limits.

Important Information

The safety shutter should be located between the laser and the input aperture of the 3-Axis system. This is the user's responsibility. Use of controls, adjustments, or procedures other than those specified in this manual without consulting a competent safety professional may result in component damage, and/or exposure to potential hazards. Always follow established industrial safety practices when operating equipment.

This system is designed to be operated in conjunction with a laser. Therefore, all applicable rules and regulations for safe operation of lasers must be known and applied when installing and operating the system. Since Novanta Inc. has no influence over the employed laser or the overall system, the customer is solely responsible for the laser safety of the entire system.

1.4 CUSTOMER SUPPORT

Before contacting Novanta for assistance, review appropriate sections in the manual that may answer your questions.

After consulting this manual, please contact one of our worldwide offices between 9 AM and 5 PM local time.

Americas, Asia Pacific

Novanta Headquarters, Bedford, USA

Phone: +1-781-266-5700

Email: photonics@novanta.com

Europe, Middle East, Africa

Novanta Europe GmbH, Wackersdorf, Germany

Phone: +49 9431 7984-0

Email: photonics@novanta.com

Milan, Italy

Phone: +39-039-793-710

Email: photonics@novanta.com

Important Information

China

Novanta Sales & Service Office, Shenzhen, China

Phone: +86-755-8280-5395

Email: photonics.china@novanta.com

Novanta Sales & Service Office, Suzhou, China

Phone: +86-512-6283-7080

Email: photonics.china@novanta.com

Japan

Novanta Service & Sales Office, Tokyo, Japan

Phone: +81-3-5753-2460

Email: photonics.japan@novanta.com

2 INTRODUCTION

2.1 GENERAL NOTES

Novanta reserves the right to make changes to the products covered in this manual to improve performance, reliability, or manufacturability.

Although every effort has been made to ensure accuracy of the information contained in this manual, Novant assumes no responsibility for inadvertent errors. Contents of the manual are subject to change without notice.

2.2 USING THIS MANUAL

2.2.1 PURPOSE

This manual covers the XML based application programming interface for the ScanMaster Controller (also known as the SMC). Information on the SMC hardware can be obtained from the *SMC Hardware Reference Manual* (Lit. No. P0900-0168).

Additional detailed operational information is contained in “1

Appendix A - Theory of Operation” on page 280.

2.2.2 REVISION HISTORY

The following table shows the revision history for this document.

Table 1 - REVISION HISTORY		
REV	DATE	Changes from previous revision
A	Dec 11, 2014	First release
B	Nov 2, 2015	<p>Attach/Detach session is now obsolete</p> <p>Redistributables directory has been reorganized</p> <p>Win32 access is through new DLL interfaces</p> <p>Added AdminConfig items to configure IP addressing</p> <p>Added ControlConfig items for marking mode control, digital polarity control, and external pause control.</p> <p>JumpAndFireList methods added to API to permit binary data passing for efficiency</p> <p>Section added for via-hole drilling explaining closed-loop and the new open-loop modes</p> <p>All Rev 1 tags in LaserConfig file have been deprecated and removed from the definitions</p> <p>Rev 2 tags <i>LaserModType</i> and <i>LaserModSyncSrc</i> are now deprecated in the LaserConfig file replaced by bits in the <i>LaserModeConfig</i> tag value</p> <p>Remote Control API now supported</p> <p>SMC LastError code table added</p> <p>MotfEnable description expanded to describe Continuous and Continuous with edge-of-field detection modes.</p>

Introduction

C	Dec 5, 2016	<p>Updated contact information and changed CTI references to Cambridge Technology</p> <p>Installation folder root has changes from “CTI” to “Cambridge Technology”</p> <p>Correction table port mappings were fixed to match the implementation</p> <p>SetMotfEncoderRate (21) Remote Control API Command is now Obsolete</p> <p>Attach/Detach session is reinstated</p> <p>C++ interfaces have been added method descriptions</p> <p>New class-based C++ wrapper DLLs are available for C++ app development.</p> <p>Added <Set id='XY2AddressMode'> command</p> <p>MotfWaitForTrigger count value is raw encoder counts</p> <p>Added JumpAndDrillList commands</p> <p>LensConfig table offsets are now I33</p> <p>Use new CT logo and fixed Japan support e-mail address</p> <p>Added a section on EC1000 Win32 application migration to the new Win32 DLLs</p> <p>Exception code table updated</p> <p>Fixed some formatting issues</p> <p>Added sendCorrectionData (overload 3)</p> <p><LaserEnable> changed to <set id='EnableLaser'></p> <p>User Config offset values are floats</p>
D	Mar 27, 2017	<p>Priority message “Restart” is now marked as available</p> <p>Remote API message “SetMotfEncoderRate” has be un-obsolete</p> <p>Footer Copyright notice changed to 2017</p>
E	July 18, 2017	<p>Added WobbleMode to set constant overlap wobble at full mark speed</p>

F	January 2018	<p>Updated description of StartupJob in the Control Config file.</p> <p>Removed obsolete AxisDACRange and ServoConfig settings in the Control Config file.</p> <p>Added HeadOffset property</p> <p>RequestFixedDataList → getFixedDataList to match the implementation</p> <p>Fixed COM port numbering to match the hardware labeling</p> <p>Added startup error codes to LastError broadcast parameter description</p> <p>Added description of extended MotfEnable modes for continuous tracking</p> <p>Fixed <set id='XY2AddressMode'> syntax</p> <p>Added pixel data description to RasterLine</p> <p>Fixed syntax error in GetCalFactors priority message</p> <p>Added <set id='SMCInsGenMode'></p> <p>Added <GSBusDisable></p>
G	June 2018	<p>Fixed typo in CalibrateJumpTime</p> <p>Changed valid range of offset values of RemoteAPI "SetPerformanceGlobals" command to be 24 bits.</p>
H	Feb 2019	<p>Added description of new Remote API command syntax</p> <p>Added ScanScript Remote API commands</p> <p>Added SyncFileSystem, StartLogging, StopLogging & PowerScale priority messages</p> <p>Added LissajousWobble, WobbleTable commands, and Wobble direction</p>

J	January 2020	<p>Removed RasterModes 2 & 3 and optimize option: unsupported modes</p> <p>Added access to both MOTF frequency values in GetRegisters</p> <p>Updated error code tables</p> <p>Message event code tables clarified</p> <p>Added explanation of laser types 100 and greater in the laser config file</p> <p>Added clarifying description about the value of the MOTF count register</p> <p>Added EnableZCompensation and SyncMasterEnabled to ControlConfig file</p> <p>Added WaitUntilGalvoCmdDelayComp to JumpAndDrillList</p> <p>Added GalvoCmdMarker command</p> <p>Added MotfTriggerEvent command</p> <p>Remote API Admin and User PIN commands 500, 501, 512, 513 are obsoleted</p> <p>Remote API commands Take/Release Host Control (2 & 3) are unobsoleted and clarified as to effect</p>
K	July 2020	<p>Updated error code tables</p> <p>Added HeadTransform command</p>
L	March 2021	<p>Added L3_INPOS signal to WaitForIO and CurrentDIO register bits</p> <p>Added Polygon Raster section</p>
M	February 2022	<p>Added priority message "WriteDigital" which was missed in V3.0 doc update</p> <p>Raster line pixel definition updated to reflect single-bit-per-pixel packing</p> <p>Corrected value range of LongDelay</p> <p>Reformatted document</p>

2.3 WARRANTY INFORMATION

The Customer shall examine each shipment within 10 days of receipt and inform Novanta of any shortage or damage. If no discrepancies are reported, the shipment will be considered as delivered complete and defect-free. Novanta warrants products against defects up to 1 year from manufacture date, barring unauthorized modifications or misuse. Repaired product is warrantied for 90 days after the repair is made, or one year after manufacture date - whichever is longer.

Introduction

Contact Customer Service at +1-781-266-5800 to obtain a Return Materials Authorization (RMA) number before returning any product for repair.

All orders are subject to the Terms and Conditions and Limited Warranty. Contact your local sales office for the latest version of these documents and other useful information.

Customers assume all responsibility for maintaining a laser-safe working environment. OEM customers must assume all responsibility for CDRH (Center for Devices and Radiological Health) certification.

3 SMC PRODUCT INTRODUCTION

3.1 SYSTEM DESCRIPTION

SMC is a self-contained controller that provides advanced hardware and software control technology to drive laser scanning systems. The Ethernet-connected SMC board is designed to permit remote embedding and control of a scan-head and laser system. It is capable of controlling two scan-heads with up to three motion axes each with concurrent laser timing control. It also provides integrated synchronization I/O for connection to factory automation equipment.

Connection to a PC for job download and administrative control is made via Ethernet® network using industry standard TCP/IP protocols. In addition to Ethernet connectivity, the SMC provides external USB connections to support job file distribution via industry standard USB Flash drives. RS232 and RS485 Serial I/O is also provided for laser control, external automation control, and diagnostic access.

In a typical installation, the SMC is a “smart controller” device, which can be installed remotely in a laser scanning system. Positioning vectors are organized as packets which represent an entire job, or sequential parts of a job. These packets are then sent from a networked PC to the SMC for local processing. The SMC sequentially processes these vectors in real-time and sends them to the laser steering galvo servos as digital signals. Alternatively, the job packets can be saved to FLASH memory on the SMC and the loaded for execution from there.

There is no requirement to dedicate a full-time host PC to a laser scanning system, as the SMC can process vectors while the PC is used for other purposes. In fact, one PC can support multiple SMC-based scanning systems with no loss in performance. This is due to the large amount of buffer memory available on the controller, the use of a separate supervisory processor on the controller to handle network communication processing, and the complete off-loading of time-critical tasks to a second real-time processor on the SMC.

Direct cabling for scan-head communication to the SMC is possible for both XY2-100-based heads and Cambridge Technology Lightning™ II heads. Laser interfacing is done through a standard 0.1” 50-pin IDC ribbon-style connector to laser personality cards or cables that present laser-specific

connections. Direct connection is also possible with sparsely populated pin-in-shell-style connectors. The laser signals are organized such that an IPG YLP fiber laser with type E interface can be directly connected using a ribbon cable.

I/O signals for automation are presented in a 0.1" 20-pin header for easy access. All I/O signals are also presented in an inter-board transition connector that can be direct connected to an expansion I/O board. This arrangement permits alternate connector usage and additional signal conditioning options.

3.2 FEATURE OVERVIEW

3.2.1 HARDWARE FEATURES

- Tethered and stand-alone operation for "embedded" installation in scanning equipment
- Dual processor architecture with integrated 100/1000BaseT Ethernet communication capability
- Real-time processing engine for precise, synchronized scanner movement and laser control
- Direct 24-bit GSBUS interface to Cambridge Technology Lightning™ II digital galvo systems
- Standard support of the 16-bit XY2-100 protocol for non-Lightning™ II heads
- Dual scan-head control via the XY2-100 or GSBUS interface
- Software-selectable polarity and timing of six TTL laser control signals
- Two auxiliary analog output channels (12-Bit) 0-10V for control of laser current or pulse intensity
- One 8-Bit TTL digital output port for laser power control
- Four 24V-compatible general purpose digital outputs
- Four 24V-compatible general purpose optically isolated digital inputs
- Seven 24V-compatible dedicated outputs and optically isolated inputs for system control and external equipment synchronization
- One USB socket and one USB header for portable flash disk access
- 3GBytes of on-board Micro SD card flash for storage of firmware, local jobs, and parameters
- 300MB RAM for downloadable job data storage
- One RS232 serial port for console and smart-display use

- One RS232 serial port for general purpose use
- One RS232 serial port for laser control (included in the laser connector)
- One RS485 serial port for smart-controller motion control
- Two quadrature encoder inputs for Mark-on-the-fly use

3.2.2 SOFTWARE FEATURES

The SMC is designed with a client-server architectural model. The SMC implements all required server code functions including the broadcast of identification and status information, vector packet handling, command and control communications, and real-time positioning operations. Host-to-SMC communications uses TCP/IP as a transport mechanism over Ethernet.

To simplify integration with third-party application software, a Microsoft Windows-compatible Application Programming Interface (API) is provided. Two API formats are supported: .NET and Win32 DLL. The APIs take care of all network connection requirements, and they abstract many of the discrete functions of the module into higher-level vector-oriented instructions.

While this document describes the low-level EC1000 compatible XML API, the recommended interface for new application development for the SMC is Cambridge Technology's high-level ScanMaster API. This API provides a high-level hardware abstraction, graphical file importing and advanced shape rendering. In addition to these features, the ScanMaster API permits access to ScanScript, the powerful embedded scripting language feature that enables flexible automation integration and local rendering of bar codes, text, and various other shapes. This capability is very useful in structuring custom applications that require real-time rendering of serial numbers and data-codes as in some mark-on-the-fly situation.

In addition to the programming interface DLLs, example code and administrative management tools are provided to facilitate setup, configuration, and calibration.

3.3 APPLICATION PROGRAMMING INTERFACE

The host software Application Programming Interface (API) is implemented in Microsoft's C# language and is exposed as Windows .NET assemblies and as COM objects. It is also accessible via a bridge DLL that provides Win32-style access without the complexity of COM. These interfaces permit access from any suitable Microsoft Windows platform programming language such as Visual Basic, C++, C#, etc.

The DLLs and .tlb files that make up the COM interface are automatically installed and registered in the Window Registry by a setup installation program on the software distribution CD. Unmanaged (non-.NET) programming languages such as C++ can access the DLLs through:

- COM objects that are imported into the IDE through the use of the COM object browser
- Traditional Win32 style wrapper DLLs

The COM interfaces are identified as ICti.Broadcast and ICti.Session. In languages based on Microsoft .NET technology, the interfaces are available as assemblies that can be referenced within a project. For backward compatibility with applications developed for the EC1000, DLLs with interfaces defined as ILecSession and ILecBroadcast are also provided however these interfaces are not recommended for continued use.

Example code that illustrates the use of the API is contained in the SDK installer and is loaded on the computer during API installation. The code examples are in a set of subdirectories in the Sample Programs directory where the API software is installed. The DLLs making up this API can also be used to control EC1000 platforms with firmware version 2.8.0 and above.

3.3.1 INSTALLATION LOCATION

The DLLs, libraries and header files that make up the API are installed in subdirectories of the following location on the installation drive (typically the C drive):

C:\Program Files (x86)\Cambridge Technology\SMC\Client \Redistributables

If the 64-bit installer is chosen, then the path will be:

C:\Program Files\Cambridge Technology\SMC\Client \Redistributables

The subdirectories Bin, Lib, and Include contain the actual files used by an application. The DLL names and their functions are defined in the following table.

Table 2 - SMC API DLLs

DLL name	Function
Cti.Broadcast.dll, Cti.Broadcast.tlb	Contains the .NET and COM Broadcast API entry points.
Cti.Session.dll, Cti.Session.tlb	Contains the .NET and COM Session API entry points.
Cti.CommonLib.dll Cti.FTPClient.dll	Contains support functions for the API. Required for use.

DLL name	Function
Cti.Session.Win32.dll, Cti.Session.Win32.lib, CTISessionWin32.h Cti.Broadcast.Win32.dll, Cti.Broadcast.Win32.lib, CTIBroadcastWin32.h, SMCEventCodes.h	Contains Win32/C++ compatible entry points to the Broadcast and Session DLLs. These interfaces are non-class based and use an optional device index argument to specify a controller in a multi-controller system.
Cti.Session.Win32Cls.dll, Cti.Session.Win32Cls.lib, CTISessionWin32Cls.h Cti.Broadcast.Win32Cls.dll, Cti.Broadcast.Win32Cls.lib, CTIBroadcastWin32Cls.h SMCEventCodes.h	Contains Win32/C++ compatible entry points to the Broadcast and Session DLLs. These interfaces are class-based and more closely match the underlying .NET interfaces. These interfaces support an unlimited number of controller connections.
Cti.TelnetClient.dll	Utility functions to support Telnet access to the SMC. Private to Cambridge Technology. Used by the firmware updater utility.
Cti.ECUtils.dll, CtiECUtilsWin32.h, Cti.ECUtilsCls.dll, CtiECUtilsWin32Cls.h	Utility functions used by the demo programs. Not necessary for normal use but contains useful functions for all applications. Source code for this DLL is provided as part of the sample programs.

3.3.2 API STRUCTURE

The API is divided into three components:

1. The Broadcast API, which is used to identify and examine the status of SMCs on the network
2. The Session API, which is used to transfer configuration and job data to and from a selected controller for real-time processing
3. The Remote Control API, which is used to provide simple ASCII character-string-level control of an SMC that has been conditioned to run locally stored marking jobs.

For convenience, the API is defined using .NET C# syntax. All functions return unsigned integer codes to indicate the success or failure of the operations. These codes are defined in *Table 37 - API Error Codes* on page 315.

The API makes extensive use of XML to pass parameters between a client application and the DLLs. This technique dramatically reduces the number of interface methods required to control an SMC module. The following sections explicitly define the XML interface requirements.

Sample programs illustrating the use of the API are located in the C:\Program Files (x86)\CTI\SMC\Client\Sample Programs directory.

3.3.3 WIN32 C++ INTERFACES

The XML API DLLs are written using Microsoft .NET technology. Two wrapper DLLs are provided to facilitate interfacing to main DLLs from unmanaged software development environments. These DLLs handle the data marshalling between the environments and can be called directly from a C++ or other unmanaged code development environments.

The interfaces are defined in the header files *CTISessionWin32Cls.h* and *CTIBroadcastWin32Cls.h* contained in the \Client\Redistributables\Include directory. Where possible, the method names and arguments are preserved intact so correlating the documentation in this manual with the method names should be straight-forward. In cases where multiple method overloads are provided in the .NET DLL, the alternate interface is differentiated with a suffix "2" at the end of the method name.

Migrating EC1000 Win32 Applications

An older deprecated C++ method interface is also provided for backwards compatibility to EC1000 based applications. This set of interfaces is not class-based and uses a device index parameter to differentiate multiple controller targets. If only a single SMC is used, then there is no need to supply the index number as it will default to zero.

These older interfaces have been repackaged into two separate DLLs with different DLL names from the EC1000 equivalent. The DLLs, link libraries and header files can be found as follows:

EC1000	SMC
\Client\EC1000Win32.dll	\Client\Redistributables\Bin\Cti.Session.Win32.dll \Client\Redistributables\Bin\Cti.Broadcast.Win32.dll
\Client\EC1000Win32.lib	\Client\Redistributables\Lib\Cti.Session.Win32.lib \Client\Redistributables\Lib\Cti.Broadcast.Win32.lib
\Client\EC1000Win32.h	\Client\Redistributables\Include\CtiSessionWin32.h

	<code>\Client\Redistributables\ Include \CtiBroadcastWin32.h</code>
--	---

The application code should be recompiled using the new header files, libraries, and DLLs. Ultimately, all the DLLs in the \Client\Redistributables\Bin directory should be copied to the customer application folder.

In the EC1000Win32.dll, there are a few undocumented utility functions that are used in some of the demo applications:

GetLocalIPAddress(...), ReadFromXMLFile(...), ReadFromXML(...) and DisplayErr(...)

These interfaces and others are exposed in a new DLL: Cti.ECUtils.Win32.dll. If the application requires these interfaces, the DLL, associated link-library and header files can be found as:

`\Client\Redistributables\Bin\Cti.ECUtils.Win32.dll`

`\Client\Redistributables\Lib\Cti.ECUtils.Win32.lib`

`\Client\Redistributables\Include\CtiECUtilsWin32.h`

4 SOFTWARE OVERVIEW

The SMC controls a laser system's galvanometers, accurately positioning deflection mirrors in synchronization with laser control signals. The sequence of motions, the speed of operation, the power that the laser uses, and the synchronization with external equipment is expressed in scanning jobs. These jobs consist of sequences of instructions to the marking engine located on the SMC module. Some instructions configure the module in such ways as setting up to emit laser control signals with the appropriate timing relative to the commanded motion of the laser steering galvos. The bulk of the instructions, however, are sequences of mark and jump instructions, which describe when and where to move the galvos and when to gate the laser control signals relative to those motions.

Job data is typically prepared using editor applications designed for that purpose. These applications may be custom software applications written by an OEM integrator, or one of several commercially available packages. Cambridge Technology's ScanMaster Designer is an example of such an application. These applications are hosted on a Microsoft Windows™-based PC and interface to the SMC modules through the API DLLs. The DLLs take care of establishing and maintaining communications with an SMC and provide a managed conduit for passing data to and from the controller. The following flowchart illustrates this arrangement.

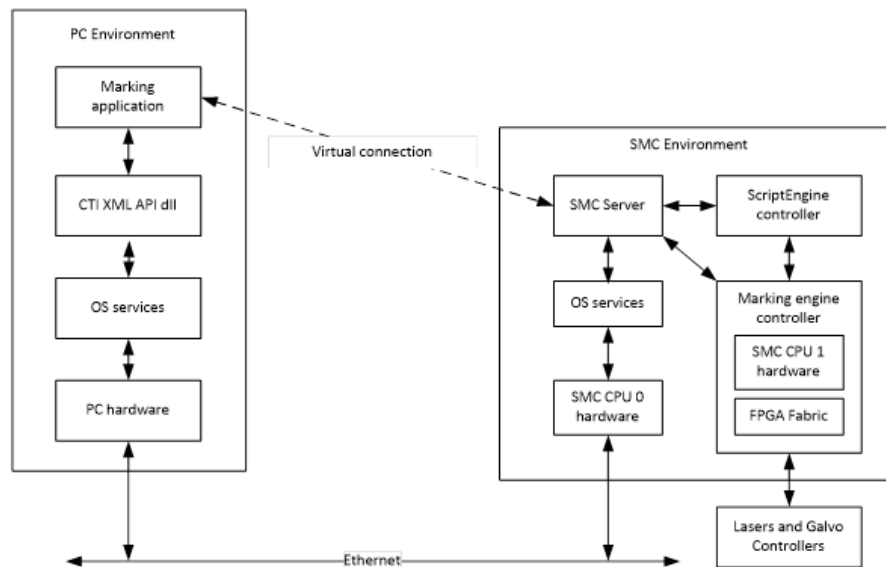
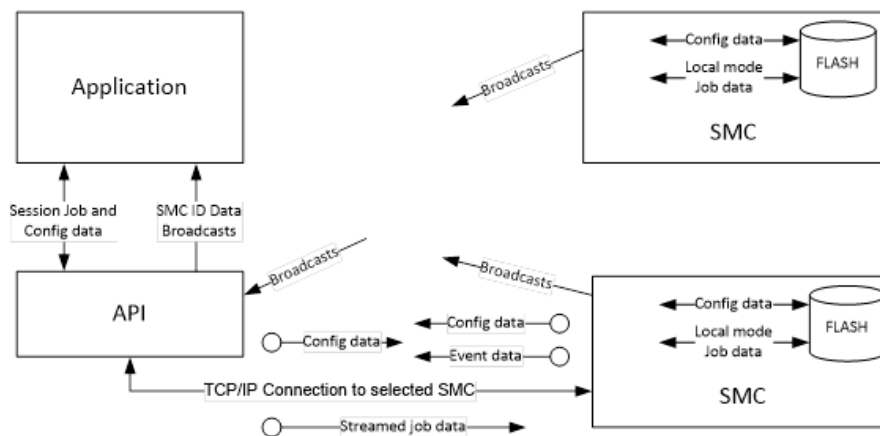


Figure 1 - CLIENT-SERVER ARCHITECTURE

The SMC contains a fully integrated processor and operating system capable of high-level communications with a supervisory host workstation using TCP/IP protocols. It can also operate in a fully independent stand-alone mode executing stored jobs. The control software of the SMC is stored in Flash memory on the module.

In a networked application, the SMC firmware boots upon system power-up and periodically broadcasts identification information on the network. Application software on a host that links with the SMC API software can accept and process these broadcast messages. The broadcast messages contain data that identifies the serial number, friendly name, and IP address of the SMC. This data, in turn is used to establish session communication channels to the controller. The following figure illustrates this relationship.

**Figure 2 - SMC SOFTWARE DATA FLOW**

A communications session permits the transmission of job data to the SMC and the reception of job-generated messages. Jobs are streamed to the SMC with multiple levels of buffering to guarantee full marking performance without CPU load-dependent timing anomalies. Two additional channels of communications are provided to permit asynchronous job aborts, job pausing and resuming, and message propagation back to the application.

The system also supports the concept of fixed configuration data (i.e., data that defines the configuration of the scan-head and surrounding electronics). Examples of such data are lens correction tables, laser interface signal polarities, lens field size, focal length, and calibration values, etc. This data can be set by a system integrator and stored in Flash memory on the SMC.

There are 2 forms of SMC API. One for use with C#, and the other for C/C++. The DLLs, header files and libraries are contained in the *Cambridge Technology\SMC\Client\Redistributables* folder and should be copied to an appropriate place in the customer's application development directory structure

4.1 THE USE OF XML IN THE API

The API uses XML syntax for setting laser timing and scanner parameters, and for specifying motion vector sequences at any desired speed. XML is a standard text-based specification language used in many internet applications to represent data in a portable manner. Documentation on XML is available from many on-line sources.

Job commands and configuration data elements can take multiple arguments to specify their function. In addition, data may be numeric of several different types or text strings. Depending on the command, parameters may be passed as XML attributes or as tag values. Lists of values are separated using a comma (",") or semi-colon (";"). Where lists of floating-point values are passed, the semi-colon separator is preferred to avoid problems with internationalization of the comma character as a decimal place specifier. The following table shows a few samples of how XML is used in the API. Example data is shown in **bold** font.

Table 3 - SAMPLE XML STATEMENTS

XML Statement	Meaning
<code><set id='JumpDelay'>200</set></code>	The "set" command is used to specify parameters that modify the behavior of a job when it is run.
<code><MarkAbsEx>1000; 2000; 300</MarkAbsEx></code>	Draw a marking vector from the current position to the target location specified. The coordinate units can be bits (default) or in user units of mm, inch, or mils provided that the bits/mm calibration factors are made known to the API.
<code><JumpAbsEx>1.25; 15.5; 0.3</JumpAbsEx></code>	Jump from the current position to the target location specified in floating point numbers

	(could be ms, inch or mils units). Note the use of the semi-colon separator.
<code><LaserStandby laser='1' width='10' period='200' /></code>	Set the standby laser modulation characteristics for the LASER_MOD1 output to a pulse width of 10 laser timing ticks with a period of 200 laser timing ticks. This attribute style notation is used in the configuration files.
<code><set id='LaserStandby'>1; 10; 200</set></code>	Equivalent to the previous example except this is the form used in a job.

Details of these statements and all others are contained in the following sections.

5 BROADCAST API

The Broadcast API is a set of methods that allow a client application to identify SMC controllers on the network and to get relevant information about those controllers. On a configurable periodic basis, the SMC modules broadcast identification packets to the network. The API captures broadcast messages from all available SMC controllers and makes this information available to the client. This information is used by the client to establish a communication session with a target controller. Sessions are used to send job data to a controller and to send/receive module configuration data. The methods used in sessions are described in Section 6 Session API.

The methods of the Broadcast API return an unsigned integer as an error code. To interpret the error codes, refer to *Table 37 - API Error Codes* on page 315.

5.1 ESTABLISHING A CONNECTION

To use the broadcast facility, a connection must be made to the Broadcast API using the following methods.

5.1.1 clientAttachBroadcast

Purpose	Establishes a connection to receive broadcast messages		
Syntax	C#	uint clientAttachBroadca st (string strMulticastAddress, string strLocalAddress, int iLocalPortNumber, ref int piClientId)

	C++	uint clientAttachBroadca st (const char * strMulticastAddress, const char * strLocalAddress, int iLocalPortNumber, int & piClientId)
Arguments	strMulticastAddress	IP address to which the SMC devices are broadcasting over (224.168.100.2 – set in the Admin Config file on the SMC)	
	strLocalAddress	IP address of the host PC’s network adaptor that is connected to the SMC modules.	
	iLocalPortNumber	Port number to which the SMC devices are broadcasting over (11000 – set in the Admin Config file on the SMC)	
	piClientId	This is an id that is used in calls to other broadcast methods.	
Comments	<p>This method is used by a client application to establish a connection to the broadcast mechanism of the SMC. Once connected, a client may receive broadcast messages from all SMC modules on the network. The messages contain information about the broadcasting module including the name, internet IP address, and other relevant data. This data is retrieved through the use of <u>getBroadcastData</u>.</p> <p>strMulticastAddress and strLocalPortNumber are values that are defined in the Administration Configuration file. For more information on the Administration Configuration file, refer to Section Error! Reference source not found. (“Error! Reference source not found.”) on page Error! Bookmark not defined.</p> <p>strLocalAddress is required to differentiate which network adaptor is connected to the SMC. The source code for a sample utility function to get this information from the Windows operating system is provided in the Sample Programs directory.</p>		
See also	<u>clientAttachBroadcast</u> , <u>getServerCount</u> , <u>getServerList</u> , <u>getBroadcastData</u>		

5.1.2 [clientDetachBroadcast](#)

Purpose	Terminates the connection to the broadcast mechanism	
Syntax	C#	uint clientDetachBroadcast(int iClientId)
	C++	uint clientDetachBroadcast(int iClientId)
Arguments	iClientId	Identifier of the connection made by the application
Comments	This method is used by a client application to terminate a connection to the broadcast mechanism of the SMC.	
See also	clientAttachBroadcast	

5.2 RETRIEVING BROADCAST DATA

Several methods are provided to get information about network-attached SMC modules.

5.2.1 `getServerCount`

Purpose	Gets embedded controller device data		
Syntax	C#	uint getServerCount(int iClientId, out int piServerCount)
	C++	uint getServerCount(int iClientId, int & piServerCount)
Arguments	iClientId	Identifier of the connection made by the application	
	piServer Count	The number of SMC devices that were identified	

Comments	<p>Once a connection to the broadcast mechanism has been established, broadcast messages are then received, and a table of available modules is built by the API.</p> <p>This method returns the number of distinct SMC modules that have transmitted valid broadcast packets since the <u>clientAttachBroadcast</u> method was called.</p> <p>Because of the asynchronous and periodic nature of the broadcast transmissions, it may take some time before all SMC controllers are recognized and reported via this method. Several successive calls may yield different results until enough time has passed to account for the longest broadcast interval. The broadcast interval is configured in the Administration Configuration file. It can be changed by using the <u>requestFixedData</u> method to retrieve it and the <u>sendFixedData</u> method to update the stored copy.</p>
See also	<u>clientAttachBroadcast</u> , <u>clientDetachBroadcast</u> , <u>getServerList</u> , <u>getBroadcastData</u>

5.2.2 [getServerList](#)

Purpose	Gets embedded controller device data		
Syntax	C#	uint getServerList(int iClientId, out int piServerCount, out string pstrDeviceList)
	C++	uint getServerList(int iClientId, int & piServerCount, const char * & pstrDeviceList)
Arguments	iClientId	Identifier of the connection made by the application	
	piServerCount	The number of SMC devices that were identified	
	pstrDeviceList	The names of the SMC devices that were identified. The string returned contains an XML representation of the data.	

Comments	<p>This method returns a list of identifiers for the SMC modules for which valid broadcast packets have been received. One of the friendly names can be used in the method <code>getBroadcastData</code> to obtain more extensive identification data.</p> <p>Because of the asynchronous and periodic nature of the broadcast transmissions, it may take some time before all SMC controllers are recognized and reported via this method. Several successive calls may yield different results until enough time has passed to account for the longest broadcast interval. The broadcast interval is configured in the Administration Configuration file. It can be changed by using the <code>requestFixedData</code> method to retrieve it and the <code>sendFixedData</code> method to update the stored copy.</p> <p>The friendly name list contains an XML representation of the data. For example:</p> <pre> <DeviceList> <Device name='SMC_Alpha' ip='192.168.42.30' mac='00:50:C2:4F:A0:01' /> <Device name='SMC_Beta' ip='192.168.42.31' mac='00:50:C2:4F:A0:06' /> </DeviceList> </pre>
See also	clientAttachBroadcast , clientDetachBroadcast , getServerCount , getBroadcastData

5.2.3 `getBroadcastData`

Purpose	Gets embedded controller device data		
Syntax	C#	uint getBroadcastData (int iClientId, string strFriendlyName, int iDataType, out string pstrData)
	C++	uint getBroadcastData (int iClientId, const char * strFriendlyName, int iDataType, const char * & pstrData)
Arguments	iClientId	Identifier of the connection made by the application	

	strFriendlyName	Name of the SMC device
	iDataType	The type of SMC device data (see Section Error! Reference source not found. (“ Error! Reference source not found. ”) on page Error! Bookmark not defined.)
	pstrData	The data requested from the SMC device. The string returned contains an XML representation of the data requested by piDataType.
Comments	This function is used by a client application to retrieve various types of data related to the specified SMC module. This data is defined in the Data Types section.	
See also	clientAttachBroadcast , clientDetachBroadcast , getServerCount , getServerList	

5.3 BROADCAST DATA DEFINITIONS

Both the Broadcast and Session APIs use a data type code. See the following table (“**Error! Reference source not found.**”) to specify the data that the application is requesting or sending. This is the *iDataType* argument in the methods [getBroadcastData](#), [requestFixedData](#), and [sendFixedData](#). All data types support an XML representation of the data.

Table 4 - BROADCAST DATA TYPES

Broadcast Data Type	iDataType Value Code
System Information	0x01
Status Information	0x07

In the following data description tables, example data is shown in **bold** font. Although in XML all data is expressed as text, the actual data type interpretation is application dependent. For the SMC, all data has an expected type interpretation, thus the tables contain a column that indicates the data

type that is intended for the particular data element. The data types are identified in the following table (**"Error! Reference source not found."**).

Table 5 - DATA TYPE KEYS

Type Identifier	Type Description	Range
STR	ASCII String	<= 256 characters
U16	Unsigned 16-bit Integer	0 <-> 65535
I16	Signed 16-bit Integer	-32768 <-> +32767
U32	Unsigned 32-bit Integer	0 <-> 4,294,967,295
I32	Signed 32-bit Integer	-2,147,483,648 <-> 2,147,483,647
FLT	Floating point	IEEE 32-bit Floating Point range
BOOL	Boolean	true, false
HEX	Unsigned 32-bit integer	0x00000000 <-> 0xFFFFFFFF

All the data retrievable using the `getBroadcastData` method is read-only.

5.3.1 BROADCASTED SYSTEM INFORMATION

The broadcasted system information data contains device, hardware, and connection information.

Note: This data defines the basic characteristics of the controller, especially as required to properly communicate with the controller. It contains a combination of live dynamic data and static data that is stored on the Flash memory of the device. All data is read-only.

See also `getBroadcastData`.

Table 6 - BROADCASTED SYSTEM INFORMATION

XML Tag	Type	Description/XML Example
Data	N/A	XML Example: <Data type='SysInfoData' rev='1.0'>

XML Tag	Type	Description/XML Example
MSN	STR	Unique board manufacturing code XML Example: <MSN> SMC-14497864 </MSN>
PVer	STR	Version of the SMC platform operating system software XML Example: <PVer> Petalinux v2 </PVer>
AVer	STR	Version of the SMC server firmware XML Example: <AVer> 2.3.24.14566 </AVer>
ObjExtVer	STR	Version of the SMC ScanScript engine firmware XML Example: <ObjExtVer> 2.3.24.14566 </ObjExtVer>
FPGAfirmVer	STR	Version of the FPGA firmware that is loaded XML Example: <FPGAfirmVer> 330180118 </FPGAfirmVer>
StateCode	U32	Connection status of SMC. Refer to the State Code table for a description of each state code. XML Example: <StateCode> 1 </StateCode>
LastError	I32	Last system error. For instance, 9001 represents a recent abort operation had completed. In the case of a faulty start-up of the SMC due to corrupted configuration files, this code represents the file type that has problems. If corruption is discovered, the SMC uses backup default configuration files in order to boot properly. See <i>Table 40 - LastError Code Descriptions</i> in page 319. XML Example: <LastError> 0 </LastError>
FreeTempStorage	U32	The amount of free storage in non-persistent memory in Kilo Bytes XML Example: <FreeTempStorage> 359174 </FreeTempStorage>
PermStoragePath	STR	The path to the root of persistent memory XML Example: <PermStoragePath> mnt </PermStoragePath>
FreePermStorage	U32	<i>(Reserved for future use)</i> The amount of free storage in persistent memory in Kbytes XML Example: <FreePermStorage> 3200000 </FreePermStorage>

XML Tag	Type	Description/XML Example
FreeUSBStorage	U32	<i>(Reserved for future use)</i> The amount of free storage in Kbytes on the USB Flash device (if the USB Flash device is connected) XML Example: <FreeUSBStorage> 1002200 </FreeUSBStorage>
MAC	STR	Hardware address XML Example: <MAC> 00:1e:c0:98:a0:af </MAC>
NetMask	STR	Network mask used by SMC. This value is either manually set, or it is provided by a DHCP or DNS server. XML Example: <NetMask> 255.255.255.0 </NetMask>
NetAssign	I32	Network assignment can be manual, provided by DHCP, or provided by DNS XML Example: <NetAssign> 1 </NetAssign>
IP	STR	IP address used by SMC. This value is either manually set, or provided by a DHCP or DNS server. This IP address is used in the loginSession method to connect to a specific SMC. XML Example: <IP> 192.168.100.20 </IP>
ConnectIP	STR	The client IP address that is currently connected to the SMC XML Example: <ConnectIP> 192.168.100.1 </ConnectIP>
FriendlyName	STR	Name used by the SMC. If the SMC has trouble booting up and needs to resort to using the backup configuration files (see LastError), the FriendlyName will be preceded with "BACKUP_" XML Example: <FriendlyName> SMC_Alpha </FriendlyName>
ConnectJob	STR	The job name that is currently marking XML Example: <ConnectJob> Hubble </ConnectJob>
Port	U32	The network port currently in use by the Job Session XML Example: <Port> 12200 </Port>
HSN	STR	<i>(Reserved for future use)</i> Marking head serial number. XML Example: <HSN> HEAD-0000023 </HSN>

XML Tag	Type	Description/XML Example
Data	N/A	End SysInfoData XML Example: </Data>

The following table contains a description of each state code for the SMC controller. The state code is included in the broadcasted system information. Refer to Table 6 - Broadcasted System Information (above) for more information on the broadcasted system information.

Table 7 - STATE CODE DESCRIPTIONS

State	Value	Description
Available	0	Available for connection
ClientTCP	1	Connected to network client
ClientSerial	2	Connected to serial client
ClientLocal	4	In local mode
Restarting	8	Server restarting
Waiting	16	Waiting for server startup
Pausing	32	Job paused
WaitingTCP	64	Waiting for TCP connection
NotAvailable	128	Server is in a transitional state and unavailable
Error	256	Unrecoverable error state
NotFound	512	Expected resource not found
FPGAError	1024	Unrecoverable FPGA error

The following table contains a description of each error code that may be set by the SMC controller. The error code is included in the broadcasted system information as the LastError tag value. Refer to

Table 6 - Broadcasted System Information (above) for more information on the broadcasted system information.

5.3.2 BROADCASTED STATUS INFORMATION

The broadcasted status includes the information in the following table, as maintained by the marking engine.

Note: The information in the following table represents the live status of the device. All data is read-only.

See also `getBroadcastData`.

Table 8 - BROADCASTED STATUS INFORMATION

XML Tag	Type	Description/XML Example
Data	N/A	StatInfoData identifier XML Example: <code><Data type='StatInfoData' rev='1.1'></code>
XPosAck	BOOL	Boolean passed from the X-axis galvo servo controller indicating that the servo is "settled" at the commanded position. This information is derived from the XY2-100 status return, bit position. Note that this feature is not supported by all galvo controllers. XML Example: <code><XPosAck>true</XPosAck></code>
YPosAck	BOOL	Boolean passed from the Y-axis galvo servo controller indicating that the servo is "settled" at the commanded position. Note that this feature is not supported by all galvo controllers. XML Example: <code><YPosAck>true</YPosAck></code>
XPos	I32	The value of the current ideal commanded X position prior to lens correction. XML Example: <code><XPos>-2489</XPos></code>
YPos	I32	The value of the current ideal commanded Y position prior to lens correction XML Example: <code><YPos>5510</YPos></code>

Table 8 - BROADCASTED STATUS INFORMATION

XML Tag	Type	Description/XML Example
XActPos	I32	The value of the actual X position after lens correction XML Example: <XActPos>-2489</XActPos>
YActPos	I32	The value of the actual Y position after lens correction XML Example: <YActPos>5510</YActPos>
XTemp	BOOL	This value is true if the X galvo servo indicates an over-temperature condition in the XY2-100 status word. Note that this feature is not supported by all galvo controllers. XML Example: <XTemp>>false</XTemp>
YTemp	BOOL	This value is true if the Y galvo servo indicates an over-temperature condition in the XY2-100 status word. Note that this feature is not supported by all galvo controllers. XML Example: <YTemp>>false</YTemp>
ContrlTemp	I16	The value of the temperature in Celsius of the SMC XML Example: <ContrlTemp>32</ContrlTemp>
XStatus	HEX	Inverted high-byte from the XY2-100 status return. Note that this value is galvo servo-controller specific. XML Example: <XStatus>0x31</XStatus>
YStatus	HEX	Inverted low-byte from the XY2-100 status return. Note that this value is galvo servo-controller specific. XML Example: <YStatus>0x31</YStatus>
XPower	BOOL	This value is true if any of the bits in the XStatus register are asserted. Note that this value is galvo servo-controller specific XML Example: <XPower>>true</XPower>
YPower	BOOL	This value is true if any of the bits in the YStatus register are asserted. Note that this value is galvo servo-controller specific. XML Example: <YPower>>true</YPower>

Table 8 - BROADCASTED STATUS INFORMATION

XML Tag	Type	Description/XML Example
Interlock	HEX	<p>This number represents a bitmask that encodes the current state of the system interlock switches. A "1" in the bit position means that the interlock has been opened in that position. Bits[3..0] represent the state of the signals INTERLOCK[4..1].</p> <p>XML Example: <Interlock>0x4</Interlock></p>
CurrentDIO	HEX	<p>This number represents a bitmask that encodes the current state of the system digital I/O lines:</p> <p>bits[3..0] == AUX_GPI[4..1]_ISO bit[5..4] == AUX_START_ISO, START bits[9..6] == INTERLOCK[4..1] → {LASER_STAT2, LASER_STAT1, LASER_STAT0, ABORT} bits[13..10] == AUX_GPO[4..1] bits[17..14] == JOBACTIVE, ERROR/NREADY, BUSY, LASING bits[24..18] == LASER_STAT[6..0] bit[25] == XY2_INPOS bit[26] == AUX_XY2_INPOS bit[27] == L3_INPOS</p> <p>XML Example: <CurrentDIO>0x1023</CurrentDIO></p>
JobMarker	U16	<p>This number is a copy of the current job marker data register that can be set by an application job via the JobMarker instruction.</p> <p>XML Example: <Jobmarker>35</JobMarker></p>
JobDataCntr	U32	<p>This number is a copy of the current job data counter. This counter is cleared whenever the marking engine encounters a <BeginJob> command. This counter represents the number of 32-bit data elements that the marking engine has processed since the last time this value was reset.</p> <p>XML Example: <JobDataCntr>32336</JobDataCntr></p>
Data	N/A	<p>End StatInfoData</p> <p>XML Example: </Data></p>

6 SESSION API

Once all SMCs are identified using the Broadcast API, individual controllers may be selected for subsequent communication. The Session API provides the methods to connect to a target SMC, to get and set configuration data, to send job data, and to manage asynchronous communications events generated by the controller. Concurrent access to multiple SMCs on a network is supported by creating multiple SMC session objects and separately logging into each one. Only one host application at a time can be logged into an SMC.

The methods of the Session API return an unsigned integer as an error code. refer to *Table 37 - API Error Codes* on page 315.

6.1 ACCESS TO SMC MODULES

6.1.1 loginSession

Purpose	Connects to an SMC device by establishing a session		
Syntax	C#	uint loginSession(string strLocalAddress, string strRemoteAddress, int iRemotePortNumber, string strUsername, string strPassword, uint uiTimeout)
	C++	uint loginSession(const char * strLocalAddress, const char * strRemoteAddress, int iRemotePortNumber, const char * strUsername, const char * strPassword, uint uiTimeout)

Arguments	strLocalAddress	IP address of the local network adaptor that is connected to the SMC modules
	strRemoteAddress	TCP/IP Address of the SMC to login. This is the "ip" attribute of the SMC selected by the application and identified in the getServerList data
	iRemotePortNumber	Network Port on the SMC supporting the session. This is the <Port> value of the SysInfoData returned from the getBroadcastData call for the selected SMC.
	strUsername	(Reserved for future use)
	strPassword	(Reserved for future use)
	uiTimeout	Duration for attempting call in seconds
Comments	<p>Once SMC modules have been identified via the use of Broadcast API, a communications session can be opened between the client and a selected target SMC. Sessions are established via a call to this method. Multiple sessions to different target SMC controllers are made by instantiating separate Session objects. A target SMC controller may only serve one client session at a time.</p> <p>strLocalAddress is required to differentiate which network adaptor is connected to the SMC. The source code for a sample utility function to get this information from the Windows operating system is provided in the Sample Programs directory.</p>	
See also	logoutSession , requestFixedData , sendFixedData , sendStreamData (overload 1) , sendPriorityData	

6.1.2 [logoutSession](#)

Purpose	Disconnects an SMC device session	
Syntax	C#	uint logoutSession(uint uiTimeout)
	C++	uint logoutSession(uint uiTimeout)

Arguments	uiTimeout	Duration for attempting call in seconds
Comments	<p>When session communication is completed, the client closes the session via a call to this method. Once the session is closed, another new session may be opened to the same or other SMC devices via a call to loginSession.</p> <p>Note that if a job was streamed out to the SMC and was still executing when the logout was invoked, the job will be immediately aborted.</p>	
See also	loginSession	

6.2 CONFIGURATION DATA MANAGEMENT

The SMC has the ability to store a large amount of data in non-volatile Flash memory. This data can be configuration data or job data. Configuration data is classified as "fixed" data (i.e. it has a lifetime that spans boot-up cycles of the controller). Some of the configuration data is set at the factory and is considered permanent read-only information. Other data is used by the controller at boot-up to properly initialize the hardware interfaces, and still other data is provided for the convenience of the application programmer to indicate the capabilities of the integrated system. All configuration data is defined in Section 6.3 (Configuration Data Definitions).

Several XML data files make up the configuration data in a hierarchical relationship as shown in the following figure:

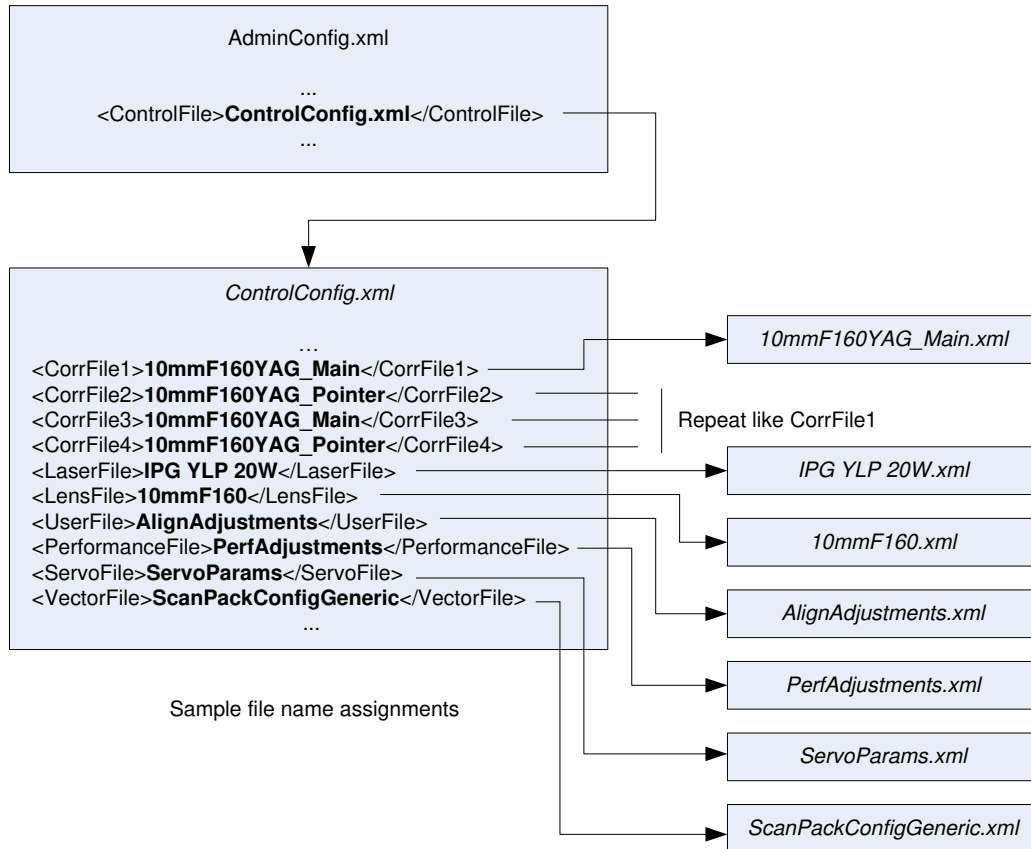


Figure 3 - SMC CONFIGURATION FILE RELATIONSHIPS

6.2.1 getFixedDataList

Purpose	Retrieves a list of the configuration files stored on the SMC		
Syntax	C#	uint getFixedDataList(out string pstrData int uiTimeout)
	C++	uint getFixedDataList(const char * & pstrData int uiTimeout)
Arguments	pstrData	Requested data	
	uiTimeout	Duration for attempting call in seconds	

Comments	<p>The returned string is in XML format. For example:</p> <pre> <FixedDataList rev='1.0'> <FixedDataType id='AdminData'> <File>AdminConfig.xml</File> </FixedDataType> <FixedDataType id='ControlConfigData'> <File>ControlConfig.xml</File> </FixedDataType> <FixedDataType id='LaserConfigData'> <File>LaserGeneric.xml</File> <File>SPI G3-HS-20.xml</File> <File>SYNRAD CO2.xml</File> </FixedDataType> <FixedDataType id='LensConfigData'> <File>LensGeneric.xml</File> <File>Lens_50mm_Co2_300mm_CF216.xml</File> </FixedDataType> <FixedDataType id='CorrTableData'> <File>50mm_Co2_300mm_CF216.xml</File> <File>PointerFinal_CF180_ZCF160.xml</File> </FixedDataType> <FixedDataType id='UserCofigData'> <File>UserGeneric.xml</File> </FixedDataType> <FixedDataType id='PerformanceMatrixData'> <File>GlobalConfigGeneric.xml</File> </FixedDataType> </FixedDataList> </pre>
See also	requestFixedData , sendFixedData

6.2.2 requestFixedData

Purpose	Retrieves fixed data from an SMC device session		
Syntax	C#	uint requestFixedData(int iDataType string strStorageName out string pstrData uint uiTimeout)
	C++	uint requestFixedData(int iDataType const char * strStorageName const char * & pstrData uint uiTimeout)
Arguments	iDataType	Identifier of the requesting data. See Table 9 - Fixed Data Type Codes on page 43.	
	strStorageName	<p>File name of the data file. The file path is constructed by the API as follows:</p> <p><code>/<PermStoragePath>/SMC/Config/<pstrStorageName>.xml</code></p> <p>where <code><PermStoragePath></code> is defined in the SysInfoData for the selected SMC and <code>pstrStorageName</code> is the name of the selected fixed data file as stored on the SMC without the ".xml" extension.</p>	
	pstrData	Requested data	
	uiTimeout	Duration for attempting call in seconds	

Comments	<p>SMC modules are autonomous devices that contain information that configures the module at boot-up for the particular hardware arrangement of the marking head. This information defines such things as the laser interface, the lens characteristics, and the optical system correction tables. An application can access this information by specifying the data type using the <code>piDataType</code> argument and providing a file name for the data as stored on the SMC. The information is returned as an XML string which must be decoded by the application. The XML specification for the different data types is defined in Section Error! Reference source not found. (“Error! Reference source not found.”) on page Error! Bookmark not defined.</p> <p>The <code>AdminConfig.xml</code> data file (see Administration Configuration) contains the element definition <code>ControlFile</code> naming the master SMC configuration file. Within this file are element definitions naming the currently active lens, laser, correction table, and user definitions files. These names are typically used as the <code>pstrStorageName</code> argument above, although other files may be accessed on the SMC file system if those file names are known and the files are of the proper type.</p>
See also	<code>getFixedDataList</code> , <code>sendFixedData</code>

6.2.3 `sendFixedData`

Purpose	Sends fixed data to an SMC device for storage		
Syntax	C#	<code>uint sendFixedData(</code>	<code>string strData</code> <code>string strStorageName</code> <code>uint uiTimeout</code>
	C++	<code>uint sendFixedData(</code>	<code>const char * strData</code> <code>const char * strStorageName</code> <code>uint uiTimeout</code>
Arguments	<code>strData</code>	The data sent to the SMC device. The string supplied contains an XML representation of the data.	

	strStorageName	File name of the data file. The file path is constructed by the API as follows: <PermStoragePath>\SMC\Config\<pstrStorageName>.xml where <u>PermStoragePath</u> is defined in the SysInfoData for the selected SMC and pstrStorageName is the name of the selected fixed data file as stored on the SMC without the ".xml" extension.
	uiTimeout	Duration for attempting call in seconds
Comments	<ul style="list-style-type: none"> • Data retrieved via the <u>requestFixedData</u> method may be modified and passed back to the controller for local storage. That data will then be immediately used and also the next time the module is booted. • An application should wait for the application message event "<u>FixedDataProcessed</u>" to be assured the updated data has been processed by the SMC and is ready for subsequent actions. 	
See also	<u>requestFixedData</u> , <u>getFixedDataList</u>	

6.3 CONFIGURATION DATA DEFINITIONS

The Session API uses a data type code to specify the data that the application is requesting or sending. This is the piDataType argument in the methods requestFixedData and sendFixedData. All data types support an XML representation of the data.

Table 9 - FIXED DATA TYPE CODES

Fixed Data Type	Data ID
Controller Configuration	0x05
Laser Configuration	0x06

Table 9 - FIXED DATA TYPE CODES

Fixed Data Type	Data ID
Lens Configuration	0x02
Correction Table	0x0D
User Configuration	0x0F
Performance Adjustments	0x10
Admin Configuration	0x0A
Servo Parameters	0x20
ScanPack Configuration	0x21

In the following data description tables, example data is shown in **bold** font. Although in XML all data is expressed as text, the actual data type interpretation is application-dependent. For the SMC, all data has an expected type interpretation, thus the tables contain a column that indicates the data type that is intended for the particular data element. The data types are identified in Table 5 – Data Types Keys.

All data that can be retrieved with the [requestFixedData](#) method is changeable with the [sendFixedData](#) method. This powerful interface permits full configurability of the SMC. Most of the elements in the data tables are set by a system integrator to provide information for a marking application programmer to configure the user-interface and control interfaces as a function of the controller/system hardware configuration. This data is not intended to be changed after it has been set by an integrator.

In addition to the integrator data, there is a table of data that is intended to be set by a system administrator. This data can be adapted at the end-customer site to meet specific networking requirements. This data is also intended to be read-only from a marking application perspective.

Some of the properties defined in the configuration data tables are provided as a convenience to the application programmer in adapting the software for various target configurations. These properties are shown first in the tables and identified with the heading “Host application initialization settings”. The properties are ignored by the controller at boot-up.

The other data in the tables identified with the heading “Hardware initialization settings” are used by the controller at boot-up to configure the laser control signals and other hardware features.

All of the configuration data is persistent on the controller and changeable via the API.

6.3.1 ADMINISTRATION CONFIGURATION

Administration Configuration data defines the base behavior of the module. Most of the items defined here are used to configure the network parameters and diagnostic tracing of the server software. The ControlFile tag defines the name of the controller configuration file which contains pointers to other files that define the configuration of the module.

The administration configuration describes configurable properties of the SMC device related to system administration.

These properties control how the SMC identifies itself and how it records tracing information about network transactions. All of these properties are used by the controller at boot-up.

Table 10 - ADMINISTRATION CONFIGURATION DATA

XML Tag	Type	Description/XML Example
Data	N/A	Begin AdminData file type data <Data type='AdminData' rev='3.0'>
DataChannel	N/A	Begin the Data Channel specification section <DataChannel>
Port	U32	The TCP/IP port number used to pass job and fixed data to and from the SMC XML Example: <Port>12200</Port>
ControlFile	STR	File name of the controller config data XML Example: <ControlFile> ControlConfig.xml </ControlFile>
EnableStreamToFile	BOOL	If True, streaming job data is sent to the <LogFile>. Used only for system debugging. XML Example: <EnableStreamToFile> False </EnableStreamToFile>

Table 10 - ADMINISTRATION CONFIGURATION DATA

XML Tag	Type	Description/XML Example
StreamFile	STR	Name of a file that will capture data streamed to the device. Used only for system debugging. XML Example: <StreamFile>LogFile.txt</StreamFile>
DataChannel	N/A	End of the Data Channel section </DataChannel>
PriorityChannel	N/A	Begin the Priority Channel specification section <PriorityChannel>
Port	U32	The TCP/IP port number used to pass priority command data to the SMC XML Example: <Port>12201</Port>
PriorityChannel	N/A	End of the Priority Channel section XML Example: </PriorityChannel>
EventChannel	N/A	Begin the Event Channel specification section XML Example: <EventChannel>
Port	U32	The TCP/IP port number used to pass event data from the SMC back to the host XML Example: <Port>12202</Port>
EventChannel	N/A	End of the Event Channel section </EventChannel>
AliveChannel	N/A	Begin the Alive Channel specification section <AliveChannel>
Port	U32	The TCP/IP port number used to pass heart-beat information between the SMC and the host XML Example: <Port>12203</Port>
AliveChannel	N/A	End of the Alive Channel section </AliveChannel>

Table 10 - ADMINISTRATION CONFIGURATION DATA

XML Tag	Type	Description/XML Example
BroadcastChannel	N/A	Begin the Broadcast Channel specification section <BroadcastChannel>
Address	STR	IP address used for broadcast messages XML Example: <Address>224.168.100.2</Address>
Port	U32	The port number used for broadcast messages XML Example: <Port>11000</Port>
Retransmit	U32	Broadcast period for the SysInfoData packet (sec) XML Example: <Retransmit type='SysInfoData' time='5'/>
Retransmit	U32	Broadcast period for the StatInfoData packet (sec) XML Example: <Retransmit type='StatInfoData' time='5'/>
BroadcastChannel	N/A	End of the Broadcast Channel section </BroadcastChannel>
Settings	N/A	Begin the miscellaneous configuration settings section. Note that the COM port assignments below must not be duplicated and must be in the range of COM0 to COM3 <Settings>
FriendlyName	STR	The friendly name given this system XML Example: <FriendlyName>SMC_Alpha</FriendlyName>
HeadSerialNumber	STR	Serial number of the head assigned by the OEM XML Example: <HeadSerialNumber>XYZ</HeadSerialNumber>
LocalMode	BOOL	The controller is to operate in local stand-alone mode on power-up. Pendant interactions are required to enable network operations. XML Example: <LocalMode>false</LocalMode>

Table 10 - ADMINISTRATION CONFIGURATION DATA

XML Tag	Type	Description/XML Example
BreakOK	BOOL	(Reserved for future use) XML Example: <BreakOK> false </BreakOK>
Client	STR	Selects the primary interface for accepting control information. Valid clients are: <div> <div>LANStream</div> <div>LAN-based streaming job control</div> </div> <div> <div>LAN</div> <div>LAN-based remote control</div> </div> <div> <div>RS232</div> <div>RS232-based remote control</div> </div> XML Example: <Client>LANStream</Client>
Pendant	STR	(Reserved for future use). XML Example: <Pendant></Pendant>
PendantPort	STR	<i>(Reserved for future use).</i> Selects the COM port used for the pendant. XML Example: <PendantPort> COM1 </PendantPort>
PendantPortSpeed	U32	<i>(Reserved for future use).</i> Baud rate for the pendant COM port. XML Example: <PendantPortSpeed> 38400 </PendantPortSpeed>
APIPort	STR	Selects the COM port used for remote API access. If the port is not specified, then no serial remote API support is available. XML Example: <APIPort> COM2 </APIPort>
APIPortSpeed	U32	Baud rate for the API COM port XML Example: <APIPortSpeed> 38400 </APIPortSpeed>
MotionPort	STR	<i>(Reserved for future use).</i> Selects the COM port used for external motion control access. If the port is not specified, then no serial motion control is available. XML Example: <MotionPort> None </MotionPort>

Table 10 - ADMINISTRATION CONFIGURATION DATA

XML Tag	Type	Description/XML Example
MotionPortSpeed	U32	<i>(Reserved for future use)</i> . Baud rate for the motion control COM port XML Example: <MotionPortSpeed> 38400 </MotionPortSpeed>
LaserPort	STR	Selects the COM port used for laser communication. If the port is not specified, then no serial laser control is available. XML Example: <LaserPort> COM3 </LaserPort>
LaserPortSpeed	U32	Baud rate for the laser COM port XML Example: <LaserPortSpeed> 38400 </LaserPortSpeed>
DFMPort	STR	<i>(Reserved for future use)</i> . Selects the COM port used for changing the position of the Dynamic Focusing Module in scan-heads equipped with this option. If the port is not specified, then no positioning control is available. XML Example: <DFMPort> None </DFMPort>
DFMPortSpeed	U32	<i>(Reserved for future use)</i> . Baud rate for the DFM positioner COM port XML Example: <DFMPortSpeed> 9600 </DFMPortSpeed>
DebugPort	STR	<i>(Reserved for future use)</i> . If assigned to a free COM port, the firmware will print debug trace messages on that port. If the port is not specified, then no debug messages are available. XML Example: <DebugPort> None </DebugPort>
DebugPortSpeed	U32	<i>(Reserved for future use)</i> . Baud rate for the software debug COM port XML Example: <DebugPortSpeed> 38400 </DebugPortSpeed>
User	STR	<i>(Reserved for future use)</i> . Password for accessing user-level pendant functions: six numeric characters only. XML Example: <User>123456</User>

Table 10 - ADMINISTRATION CONFIGURATION DATA

XML Tag	Type	Description/XML Example
Admin	STR	<i>(Reserved for future use)</i> . Password for accessing administrator-level pendant functions: six numeric characters only. XML Example: <Admin>654321</Admin>
LoggingLevel	U32	<i>(Reserved for future use)</i> . Level of transaction logging to perform; used only for system debugging. XML Example: <LoggingLevel>0</LoggingLevel>
IPMode	STR	Defines the behavior of the TCP/IP system. Values are: Static Use the IP Address, Subnet, and Gateway values below Autodetect IP information comes from a DHCP server XML Example: <IPMode>Static</IPMode>
IPAddress	STR	Use this IP Address if IPMode is set to Static XML Example: <IPAddress>192.168.100.20</IPAddress>
IPSubnet	STR	Use this IP Subnet mask if IPMode is set to Static XML Example: <IPSubnet>255.255.255.0</IPSubnet>
IPGateway	STR	Use this IP Gateway address if IPMode is set to Static XML Example: <IPGateway>192.168.100.1</IPGateway>
IPTimeout	U32	If IPMode is Autodetect, the server will wait this long in seconds for an address to be assigned by a DHCP server. If IPRetries has reached the specified limit, the static default IP Address 192.168.100.20 will be used. XML Example: <IPTimeout>10</IPTimeout>
IPRetries	U32	Numer of time to query the DHCP server for an IP address before giving up. XML Example: <IPRetries>3</IPRetries>
IPTryagain	U32	Numer of time to query the DHCP server for an IP address before giving up. XML Example: <IPTryagain>20</IPTryagain>

Table 10 - ADMINISTRATION CONFIGURATION DATA

XML Tag	Type	Description/XML Example
Settings	N/A	End Settings </Settings>
Data	N/A	End AdminData </Data>

6.3.2 CONTROLLER CONFIGURATION

The Controller Configuration file is the master control file for defining the startup configuration of the controller. It contains pointers to other configuration files that deal with specific elements of the system such as laser timing, correction tables, lens identification, user adjustments, etc. The file names referenced in the table are XML file names with the .xml extension suppressed. The files are in the /<PermStoragePath>/SMC/Config directory on the SMC. <PermStoragePath> is the value reported in the broadcasted SystemData packets.

The values in the Controller Configuration file are normally set by the integrator and are not intended to be altered by a marking application.

Note: When the Controller Configuration is sent to the SMC, the correction table and laser configurations referenced are also applied to the controller. Detailed MOTF operation is controlled through instructions passed as part of the job stream and is not a "mode" of the controller.

See also requestFixedData and sendFixedData.

Controller Configuration Data

The Following table contains the setting for the Controller Configuration file.

Table 11 - CONTROLLER CONFIGURATION DATA

XML Tag	Type	Description/XML Example
Data	N/A	Begin Controller Configuration file data <Data type='ControlConfigData' rev='3.1'>
MotfCapable	BOOL	<i>(Reserved for future use).</i> System is Mark-On-The-Fly (MOTF) capable (true). XML Example: <MotfCapable>true</MotfCapable>
MotfCalGain	FLT	<i>(Reserved for future use).</i> MOTF digital gain factor; used as a fine-tuning scalar adjustment of <u>MotfCalFactor</u> . XML Example: <MotfCalGain>1.0</MotfCalGain>
CorrFile1	STR	The name of correction table 1 file XML Example: <CorrFile1>CORRTAB1</CorrFile1>
CorrFile2	STR	The name of correction table 2 file XML Example: <CorrFile2>CORRTAB2</CorrFile2>
CorrFile3	STR	The name of correction table 3 file XML Example: <CorrFile3>CORRTAB3</CorrFile3>
CorrFile4	STR	The name of correction table 4 file XML Example: <CorrFile4>CORRTAB4</CorrFile4>
LensFile	STR	The name of the lens configuration file XML Example: <LensFile>LENSFILE2</LensFile>
LaserFile	STR	The name of the laser configuration file XML Example: <LaserFile>LASERFILE4</LaserFile>
UserFile	STR	The name of the user configuration file XML Example: <UserFile>MYCONFIGFILE</UserFile>
PerformanceFile	STR	The name of the performance adjustments file XML Example: <PerformanceFile>PADJUST</PerformanceFile>
ServoFile	STR	The name of the file that contains parameters of the galvo/servo system attached to the SMC. Used in adjusting the dynamic behavior of

Table 11 - CONTROLLER CONFIGURATION DATA

XML Tag	Type	Description/XML Example
		the embedded ScanPack algorithms to match the galvo/servo capability. XML Example: <ServoFile> ServoParams </ServoFile>
VectorFile	STR	The name of the file that contains default shape parameters for the embedded ScanPack algorithms. XML Example: <VectorFile>ScanPackConfigGeneric</VectorFile>
MotfEncoderCal	FLT	MOTF calibration factor. Relates the encoder counts to laser positioning bits (bits/count). XML Example: < MotfEncoderCal >24.23</ MotfEncoderCal >
MotfMode	U16	MOTF operational mode: 0 - Use encoder 1 - Simulate encoder XML Example: <MotfMode>0</MotfMode>
MotfDirection	I16	MOTF orientation and direction in degrees: 0 - left to right in the X-axis 90 - bottom to top in the Y-axis 180 - right to left in the X-axis 270 - Top to bottom in the Y-axis XML Example: <MotfDirection>0</MotfDirection>
LaserPipelineDelay	U16	The time in laser timing ticks that all laser signals are delayed relative to micro-vector generation. This is used to compensate for the inherent delay in servo modules from when a command is applied to when the galvos actually respond. Units are micro-seconds. XML Example: <LaserPipelineDelay>450</LaserPipelineDelay> The maximum pipeline delay value is equivalent to 4000 laser ticks so the specified value maximum will be reduced depending on the <u>LaserTiming</u> value. For example, if LaserTiming is 50 (1usec resolution) then the maximum value will be 4000usec. If LaserTiming is set to 5 (0.1usec resolution), then the maximum pipeline value is 400usec.
CmdRangeCheck Mode	U16	Command range checking mode:

Table 11 - CONTROLLER CONFIGURATION DATA

XML Tag	Type	Description/XML Example							
		<table><tr><td>Enable</td><td>enable checking if non-zero</td></tr><tr><td>Port</td><td>digital output port to manipulate</td></tr><tr><td>Value</td><td>port value to set if out of range</td></tr></table>	Enable	enable checking if non-zero	Port	digital output port to manipulate	Value	port value to set if out of range	
Enable	enable checking if non-zero								
Port	digital output port to manipulate								
Value	port value to set if out of range								
		<p>The port and value definitions are the same as the <u>WriteDigital</u> command.</p> <p>This command is used to assert an I/O output if the galvo command range is exceeded, usually during MOTF operations.</p> <p>XML Example: <CmdRangeCheckMode>1;3;1</CmdRangeCheckMode></p>							
IntlockConfig	HEX	<p>Interlock configuration control. In the SMC, the internal Interlock signals are an aggregate of the external signals {LASER_STAT2, LASER_STAT1, LASER_STAT0, and ABORT}</p> <p>There are two fields in the argument:</p> <table><tr><td>Polarity</td><td>Bits[3..0] represent the interlock signals INTLOCK[4..1]. A "1" corresponds to no current flowing through the interlock optical isolator. This condition is the interlock open state.</td></tr><tr><td>Enable</td><td>Bits[11..8] represent the interlock signals INTLOCK[4..1]. A "1" enables a transition of the interlock signal going from the unasserted to the asserted state to generate an "Interlock" exception and shut down an active job provided that bit 12 is also asserted. Bit[12] is the master enable bit for the interlock function. If this bit is set, then all enabled interlock signals should be de-asserted at power-up time or else an immediate "Interlock" exception will</td></tr></table>		Polarity	Bits[3..0] represent the interlock signals INTLOCK[4..1]. A "1" corresponds to no current flowing through the interlock optical isolator. This condition is the interlock open state.	Enable	Bits[11..8] represent the interlock signals INTLOCK[4..1]. A "1" enables a transition of the interlock signal going from the unasserted to the asserted state to generate an "Interlock" exception and shut down an active job provided that bit 12 is also asserted. Bit[12] is the master enable bit for the interlock function. If this bit is set, then all enabled interlock signals should be de-asserted at power-up time or else an immediate "Interlock" exception will		
Polarity	Bits[3..0] represent the interlock signals INTLOCK[4..1]. A "1" corresponds to no current flowing through the interlock optical isolator. This condition is the interlock open state.								
Enable	Bits[11..8] represent the interlock signals INTLOCK[4..1]. A "1" enables a transition of the interlock signal going from the unasserted to the asserted state to generate an "Interlock" exception and shut down an active job provided that bit 12 is also asserted. Bit[12] is the master enable bit for the interlock function. If this bit is set, then all enabled interlock signals should be de-asserted at power-up time or else an immediate "Interlock" exception will								

Table 11 - CONTROLLER CONFIGURATION DATA

XML Tag	Type	Description/XML Example	
			<p>be generated when this parameter is processed. All of the Enable bits can also be manipulated using the <u>SetInterlockEnable</u> priority data message.</p> <p>If an interlock that is enabled is tripped, the condition that caused the trip must be cleared and an "<u>Abort</u>" priority message sent before a job can be restarted without generating another "Interlock" exception. The current state of the interlock physical signals can be seen in the Broadcast Status data as element <u>Interlock</u>. XML Example: <IntlockConfig>0x1707</IntlockConfig></p>
XY2StatusTiming	STR	<p>Defines the timing of the decoding of the XY2-100 status line. Early timing means that the data is clocked on the rising edge of the clock, Late timing means that the data is clocked on the falling edge of the clock. XML Example: < XY2StatusTiming >Early</XY2StatusTiming ></p>	
XY2AddressingMode	STR	<p>Defines the command data width of the XY2-100 interface. Normal is traditional 16-bit command data. Enhanced is 20-bit command data used with Cambridge Technology Lightning-II galvos with an XY2-100 interface. XML Example: <XY2AddressingMode>Normal</XY2AddressingMode></p>	
XY2FrameRate	STR	<p>Defines the update rate of the XY2-100 digital interface in KHz. If this value is changed, not all XY2-100 based scan heads may respond properly. XML Example: <XY2FrameRate>100</XY2FrameRate></p>	

Table 11 - CONTROLLER CONFIGURATION DATA

XML Tag	Type	Description/XML Example
InsGenMode	STR	<p>Defines the command generation mode of operation of the SMC. Values are:</p> <p>Traditional -- Generate galvos command waveforms in the traditional Mark/Jump mode along with the appropriate delays.</p> <p>ScanPack – Generate galvo commands using Cambridge Technology’s proprietary ScanPack algorithms</p> <p>XML Example: <InsGenMode>ScanPack</InsGenMode></p>
MicroStepMode	STR	<p>If InsGenMode is set to Traditional, this defines how the calculated micro-step values are delivered to the output stage. Values are:</p> <p>ISR – Output rate timing is governed by the Mark/Jump speed command update rate value which is regulated using a timed interrupt service routine if ISRGenMode is set to Program.</p> <p>Free – Output values are calculated as quickly as possible and placed in an output FIFO for consumption at a rate governed by the XY2-100 or GSBUS frame sync.</p> <p>XML Example: <MicroStepMode>ISR</MicroStepMode></p>
ISRGenMode	STR	<p>If MicroStepMode is set to ISR, this defines the rate the micro-step values are delivered to the output stage. Values are:</p> <p>Program – Output rate timing is governed by the Mark/Jump speed command update rate value.</p> <p>FrameSync – Output values are calculated and consumed at a rate governed by the XY2-100 or GSBUS frame sync.</p> <p>XML Example: <ISRGenMode>Program</ISRGenMode></p>
RTCCCompatibility	BOOL	<p>If True, the X axis output of the Correction table calculation is delivered to the Y Galvo axis, and the Y output value is delivered to the X Galvo axis. This results in a 90 degree coordinate system rotation in the counter-clockwise direction which makes it compatible with Scanlab’s RTC and scan-head conventions.</p> <p>XML Example: <RTCCCompatibility>True</RTCCCompatibility></p>
InitPosition	U16	<p>Commands the galvos to jump to the position specified. This command is executed before <u>StartupJob</u> is processed. If this command is not present, then a jump to 0,0,0 will be done.</p>

Table 11 - CONTROLLER CONFIGURATION DATA

XML Tag	Type	Description/XML Example												
		XML Example: <InitPosition> 30000; 30000; 0</InitPosition>												
StartupJob	STR	<p>Name of a locally stored job to run after the controller boots up. Jobs can be Rev 1.0 style (.wlb), Rev 2.0 style (.job), or ScanMaster style (.lsj)</p> <p>XML Example: <StartupJob>HWInit.job</StartupJob></p> <p>NOTE: .lsj style jobs that have human-interaction commands should not be used as a startup job as host-based dialog-box support will not necessarily be present.</p>												
ExtPauseControl	STR	<p>This permits the specification of a set of external digital inputs that can cause the SMC to pause vector processing. Multiple pins along with a polarity setting may be specified which are evaluated in a logical OR configuration.</p> <p>XML Example:</p> <pre><ExtPauseControl> <Config pin="1" state="0" /> </ExtPauseControl></pre> <p>Pin numbering corresponds to the follow table:</p> <table><tr><td>0</td><td>AUX_START_ISO</td></tr><tr><td>4-1</td><td>GPI[4-1]_ISO</td></tr><tr><td>5</td><td>START</td></tr><tr><td>6</td><td>ABORT</td></tr><tr><td>13-7</td><td>LASER_STAT[6-0]</td></tr><tr><td>31-16</td><td>AUX_DIN[15-0]</td></tr></table>	0	AUX_START_ISO	4-1	GPI[4-1]_ISO	5	START	6	ABORT	13-7	LASER_STAT[6-0]	31-16	AUX_DIN[15-0]
0	AUX_START_ISO													
4-1	GPI[4-1]_ISO													
5	START													
6	ABORT													
13-7	LASER_STAT[6-0]													
31-16	AUX_DIN[15-0]													
DigitalIOPolarity	HEX	<p>The polarity of digital inputs and outputs can be changed in sub-groups as needed to make the XML and ScanScript job commands reflect a more natural signal control scheme. Setting the bit inverts the natural polarity of the signal. For optically isolated inputs, the natural state is asserted if open. For outputs, the signal naturally goes low if asserted. Both situations can be referred to as using negative logic.</p> <p>XML Example: DigitalIOPolarity>0x7f1f</DigitalIOPolarity></p> <p>Bit position assignments are in the following table:</p> <table><tr><td>0</td><td>AUX_ABORT</td></tr><tr><td>1</td><td>START</td></tr></table>	0	AUX_ABORT	1	START								
0	AUX_ABORT													
1	START													

Table 11 - CONTROLLER CONFIGURATION DATA

XML Tag	Type	Description/XML Example
		2 AUX_START_ISO 3 AUX_GPI[4-1]_ISO 4 EXT_AUX_GPI[15-0] 8 AUX_BUSY 9 AUX_READY 10 AUX_LASING 11 AUX_JOBACTIVE 12 LASER_STAT[6-0] 13 AUX_GPO[4-1] 14 EXT_AUX_GPO[15-0]
SyncMasterEnabled	BOOL	If true, enables SyncMaster functionality in the SMC. This feature is further regulated by licensing. Contact Cambridge Technology technical support for additional requirements XML Example: <SyncMasterEnabled>true</SyncMasterEnabled>
EnableZCompensation	BOOL	If true, geometric compensation is applied to the XY coordinates as Z is varied in the job data. This keeps the geometry of the marking area accurate as focus is adjusted to mark on 3D objects. Proper behavior of this compensation depends on accurate geometry being specified in the lens correction table file. If false, no geometric compensations are applied which results in a de-focused spot as Z is varied in the job data. XML Example: < EnableZCompensation >false</ EnableZCompensation >
Data	N/A	End Controller Configuration file Data </Data>

6.3.3 LASER CONFIGURATION

The Laser Configuration file defines the properties of the laser being used with the SMC.

The values in the Controller Configuration file are normally set by the integrator and are not intended to be altered by a marking application.

See also requestFixedData and sendFixedData.

Laser Configuration Data: Header and Host Application Initialization Settings

Table 12 - LASER CONFIGURATION DATA: HEADER AND HOST APPLICATION INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
Data	N/A	Laser Configuration file identifier and revision <Data type='LaserConfigData' rev='3.1'>
LsrName	STR	The name of the laser XML Example: <LsrName>IPC002</LsrName>
LsrType	U16	Application definable value to identify a laser type. Laser type values of 100 or greater are intended for use with pulse-width modulation lasers such as CO2 lasers. With these lasers, the pulse width duty cycle will be scaled according to the laser correction table. XML Example: <LsrType>1</LsrType>
FixedFreq	BOOL	Laser is only capable of a fixed frequency setting (true) or capable of variable frequency settings (false) XML Example: <FixedFreq>true</FixedFreq>
FixedPW	BOOL	Laser is only capable of a fixed pulse width setting (true) or capable of variable pulse width settings (false) XML Example: <FixedPW>true</FixedPW>
FixedWatts	BOOL	Laser is only capable of a fixed output power setting (true) or capable of variable output power settings (false) XML Example: <FixedWatts>true</FixedWatts>
WattsUnits	BOOL	Laser power units are in Watts (true) or % duty-cycle (false) XML Example: <WattsUnits>true</WattsUnits>
Pulse	U16	Pulse width range supported by the laser (in µsecs) XML Example: <Pulse min='2' max='65535' />

Table 12 - LASER CONFIGURATION DATA: HEADER AND HOST APPLICATION INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
Bits	U16	Binary value range for lasers with digital power control XML Example: <Bits min='0' max='255'/>
ExtPwrCtrl	BOOL	Laser power is controllable via an external knob (true) XML Example: <ExtPwrCtrl>false</ExtPwrCtrl>
UseExtPwrCtrl	BOOL	Application is configured to use external power control (true) XML Example: <UseExtPwrCtrl>false</UseExtPwrCtrl>
VisPtr	BOOL	Laser has a visible pointer integrated into it (true) XML Example: <VisPtr>false</VisPtr>
Duty	U16	Duty cycle range of the laser pulses (%) XML Example: <Duty min='1' max='90'/>
Freq	U16	Pulse frequency range sustainable by the laser (KHz) XML Example: <Freq min='1' max='250'/>
Watts	U16	Wattage range producible by the laser XML Example: <Watts min='1' max='15'/>
Volts	U16	Analog power level voltage range sustainable by the laser; the SMC is capable of 0-10 Volts output XML Example: <Volts min='1' max='10'/>
Interlock	STR	The name of a file on the host platform that contains instructions on how to clear an interlock break XML Example: <Interlock>IPCIntlocks.txt</Interlock>

Laser Configuration File: Hardware Initialization Settings

The following tables contain the hardware initialization settings for the Laser Configuration file.

Table 13 - HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example		
LaserModeConfig	U16	Set the laser configuration using a bit mask encoded as shown in the following list. Note that this command will override other commands that may set individual bits intended for this control word. Bit Value definitions are provided in the table on the next page.		
		Name	Hex Bit Value	Definition
		LASER_GATE polarity	0x0001	0=active high, 1=active low
		LASER_POINTER polarity	0x0002	0=active high, 1=active low
		Laser Sync Mode Bit 0	0x0004	See notes below.
		LASER_MOD1 polarity	0x0008	0=active high, 1=active low
		LASER_MOD2 polarity	0x0010	0=active high, 1=active low
		LASER_MOD3 polarity	0x0020	0=active high, 1=active low
		LASER_ENABLE polarity	0x0040	0=active high, 1=active low
		LASER_DOUT polarity	0x0080	0=active high, 1=active low
		Laser activate	0x0100	1=activate (enable) laser output signals

Table 13 - HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example		
		Laser Power Port mode	0x0200	(Obsolete – 7-bit mode is no longer supported) Set the mode of the digital laser power port 0=8-bit mode, 1=7-bit mode (LSB used as strobe)
		LASER_POINTER configuration	0x0800 & 0x0400	Sets the mode of operation of LASER_POINTER 0 – LASER_POINTER == NOT LASER_GATE 1 - LASER_POINTER == LASER_GATE & NOT LasersEnabled 2 - LASER_POINTER == NOT LasersEnabled 3 - LASER_POINTER == Asserted all of the time
		Laser Power Port	0x1000	0=8-bit digital power port, 1=analog output A1
		LASER_GATE configuration	0x2000	0=Gating signal, 1=Modulation signal if 8-bit digital power port bit 7 is also set

Table 13 - HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example		
		LASER_GATE inhibit	0x4000	0=normal operation, 1=LASER_GATE is suppressed when the laser is turned on but the modulation signal is still emitted. Use in synchronous laser operation during <u>JumpAndFireList</u> commands.
		Laser Sync Mode Bit 1	0x8000	See notes below.
		XML Example: <LaserModeConfig> 0x140 </LaserModeConfig> Notes on Laser Sync Mode: Laser Sync Mode bits [1 – 0] encode the laser synchronization mode of the SMC according to the following table:		
		Asynchronous modulation. The laser modulation is discontinuous, switching between the background modulation and the lasing modulation coincident with the LASER_GATE signal		
		Synchronous to the modulation signal on LASER_MOD3. LASER_MOD3 takes its modulation settings from the background settings for LASER_MOD1. The background signal for LASER_MOD1 and LASER_MOD2 is set for no modulation. In this mode, the LASER_GATE and subsequent LASER_MOD1 and LASER_MOD2 timing is synchronized to the rising edge of pulses on LASER_MOD3		
		Synchronous to the free-running modulation of LASER_MOD2. In this mode the LASER_GATE signal is synchronized to the falling edge of LASER_MOD2. Both LASER_MOD1 and LASER_MOD2 are free-running according to the LaserPulse settings defined for them.		

Table 13 - HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
		3 Synchronous to the external signal source received on LASER_STAT6. In this mode, the LASER_GATE and subsequent LASER_MOD1 and LASER_MOD2 timing is synchronized to the rising edge of pulses received on LASER_STAT6.
LaserTiming	U16	The number of 20ns intervals that make up a laser timing "tick" XML Example: <LaserTiming>50</LaserTiming> This example produces a timing resolution of 1usec meaning that laser modulation signals can be specified with a resolution of 1usec. The minimum LaserTiming value is 1 (20nsec).
LaserEnableDelay	U16	The time required (in milliseconds) for enabling the laser prior to actual use; sets the time that the signal LASER_ENABLE is asserted prior to a marking operation. XML Example: <LaserEnableDelay>10</LaserEnableDelay>
LaserEnableTimeout	U16	The time (in milliseconds) that the signal LASER_ENABLE will remain asserted after a marking operation. If a subsequent marking operation is started prior to the expiration of this time, then LASER_ENABLE will remain asserted and the marking operation will begin immediately without the cost of another LaserEnableDelay. XML Example: <LaserEnableTimeout>20</LaserEnableTimeout>
LaserModDelay	U16	The time (in μ secs) from the assertion of LASER_GATE to the emission of laser pulses XML Example: <LaserModDelay>20</LaserModDelay>
LaserFPK	I16	(μ sec) Sets the 'position' of the LASER_MOD3 signal relative to the LASER_GATE signal, and the 'width' of the LASER_MOD3 pulse XML Example: <LaserFPK position='0' width='10'/>
LaserStandby	U16	(μ sec) Sets the idle or non-lasing state modulation characteristics (pulse width and period) of the LASER_MOD1 and LASER_MOD2 signals. The 'period' value must be the same for both lasers XML Example: <LaserStandby laser='1' width='1' period='200'/> XML Example: <LaserStandby laser='2' width='1' period='200'/>

Table 13 - HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
LaserPowerDelay	U16	The time required (in milliseconds) after laser power is changed until the laser power has settled. Used when constructing jobs that manipulate the laser power. XML Example: <LaserPowerDelay>100</LaserPowerDelay>
InitAnalog	U16	Sets initial values for the analog output ports. XML Example: <InitAnalog port='0' value='50' /> port = 0 is LASER_ANALOG1 and port = 1 is LASER_ANALOG2 value ranges between 0 - 4095
InitDigital	U16	Sets initial values for the digital output ports. XML Example: <InitDigital port='102' value='128' /> port = 102 is the LASER_DATA port value ranges between 0 - 255
InitLaser	BOOL	<i>(Reserved for future use).</i> If set, use the settings specified in the tags InitType, InitStrDelim, InitStrEOL, InitStrings, DeinitStrings, to initialize the laser using a serial port connection. XML Example: <InitLaser>false</InitLaser>
InitType	U16	<i>(Reserved for future use).</i> Laser communications type: 0 = RS-232 Serial, 1 = Ethernet XML Example: <InitType>0</InitType>
InitStrDelim	CHR	<i>(Reserved for future use).</i> Delimiter character separating command and argument tokens in the InitString. XML Example: <InitStrDelim>","</InitStrDelim>
InitStrEOL	CHR	<i>(Reserved for future use).</i> Line termination character used by the laser command interpreter. XML Example: <InitStrEOL>"\n"</InitStrEOL>

Table 13 - HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
InitStrings	STR	<p><i>(Reserved for future use)</i>. A list of initialization strings to be sent to the laser. The list may be arbitrarily long.</p> <p>XML Example:</p> <pre><InitStrings> <InitString>ab</InitString> <InitString>cd</InitString> <InitString>ef</InitString> </InitStrings></pre>
DeinitStrings	STR	<p><i>(Reserved for future use)</i>. A list of de-initialization strings to be sent to the laser. The list may be arbitrarily long.</p> <p>XML Example:</p> <pre><DeinitStrings> <DeinitString>zy</DeinitString> <DeinitString>xw</DeinitString> <DeinitString>vu</DeinitString> </DeinitStrings></pre>
CorrTable		<p>A list of laser power linearization values. Laser power has a logical range of 0-255 and as a power change is requested, the logical power value is used to index this table and the selected entry is used as the actual "corrected" value. In the case of laser types 100 and greater, the values represent a duty-cycle value where the 0 represents 0% duty cycle and 255 represent 100% duty-cycle.</p> <p>XML Example:</p> <pre><CorrTable> <Entry>0</Entry> <Entry>1</Entry> ... <Entry>255</Entry> </CorrTable></pre>

6.3.4 LENS CONFIGURATION

The Lens Configuration file defines the properties of the lens being used with the SMC.

The values in the Lens Configuration file are normally set by the integrator and are not intended to be altered by a marking application.

See also requestFixedData and sendFixedData.

Lens Configuration Data: Header and Host Application Initialization Settings

The following table contains the header and host application initialization settings for the Lens Configuration file.

Note: The host application initialization settings are not required or used by the hardware. They are provided in the following table for the convenience of host application user parameter initialization.

Table 14 - LENS CONFIGURATION DATA: HEADER AND HOST APPLICATION INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example									
Data	N/A	LensConfigData identifier and revision <Data type='LensConfigData' rev='3.0'>									
LensName	STR	Used by the head integrator to identify a particular lens model. XML Example: <LensName>S4LFT0163</LensName>									
CalFlag	BOOL	Used by an application to indicate that this lens can be calibrated. XML Example: <CalFlag>>false</CalFlag>									
ZMode	U16	Specifies the Z-axis operational mode: <table border="1"> <thead> <tr> <th>Name</th><th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>2D</td><td>0</td><td>No Z-axis is present in the system and only X and Y vector data is used.</td></tr> <tr> <td>3D</td><td>1</td><td>Z-axis is present and the Z position is the Z-axis job data adjusted by the interpolated value from the Z-axis component of the currently active correction table. The Z-axis moves smoothly to the target position over the</td></tr> </tbody> </table>	Name	Value	Description	2D	0	No Z-axis is present in the system and only X and Y vector data is used.	3D	1	Z-axis is present and the Z position is the Z-axis job data adjusted by the interpolated value from the Z-axis component of the currently active correction table. The Z-axis moves smoothly to the target position over the
Name	Value	Description									
2D	0	No Z-axis is present in the system and only X and Y vector data is used.									
3D	1	Z-axis is present and the Z position is the Z-axis job data adjusted by the interpolated value from the Z-axis component of the currently active correction table. The Z-axis moves smoothly to the target position over the									

Table 14 - LENS CONFIGURATION DATA: HEADER AND HOST APPLICATION INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example		
				same time period it takes to move to the X-Y target position.
		XML Example: <ZMode>0</ZMode>		
FocalLen	U32	Focal length of the lens (mm) XML Example: <FocalLen>163</FocalLen>		
Aperture	U32	Laser beam diameter entering the lens (mm) XML Example: <Aperture>15</Aperture>		

Lens Configuration Data: Hardware Initialization Settings

The following tables contain the hardware initialization settings for the Lens Configuration file.

Note: The Tbl{1,2,3,4} offset, gain and rotation factors are intended to be used by the integrator to correct for system alignment issues and for the effects of the different wavelengths of light used for marking (table 1) and pointing (table 2). User-level adjustments to the imaging field are performed through the use of The User Configuration Table. The order of application of the factors is as follows:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} XGain \cdot \cos(Rotation) & XGain \cdot (-\sin(Rotation)) \\ YGain \cdot \sin(Rotation) & YGain \cdot \cos(Rotation) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} Xoff \\ Yoff \end{bmatrix}$$

Table 15 - LENS CONFIGURATION DATA: HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
Tbl1XOff	I32	X-axis offset to be applied to correction table 1 (bits) XML Example: <Tbl1XOff>0</Tbl1XOff>
Tbl1YOff	I32	Y-axis offset to be applied to correction table 1 (bits) XML Example: <Tbl1YOff>0</Tbl1YOff>
Tbl1XGain	FLT	X-axis gain to be applied to correction table 1 XML Example: <Tbl1XGain>1.0</Tbl1XGain>
Tbl1YGain	FLT	Y-axis gain to be applied to correction table 1 XML Example: <Tbl1YGain>1.0</Tbl1YGain>
Tbl1Rotation	FLT	Field rotation to be applied to correction table 1 (degrees) XML Example: <Tbl1Rotation>0.0</Tbl1Rotation>
Tbl2XOff	I32	X-axis offset to be applied to correction table 2 (bits) XML Example: <Tbl2XOff>0</Tbl2XOff>
Tbl2YOff	I32	Y-axis offset to be applied to correction table 2 (bits) XML Example: <Tbl2YOff>0</Tbl2YOff>
Tbl2XGain	FLT	X-axis gain to be applied to correction table 2 XML Example: <Tbl2XGain>1.0</Tbl2XGain>
Tbl2YGain	FLT	Y-axis gain to be applied to correction table 2 XML Example: <Tbl2YGain>1.0</Tbl2YGain>
Tbl2Rotation	FLT	Field rotation to be applied to correction table 2 XML Example: <Tbl2Rotation>0.0</Tbl2Rotation>
Tbl3XOff	I32	X-axis offset to be applied to correction table 3 (bits) XML Example: <Tbl3XOff>0</Tbl3XOff>
Tbl3YOff	I32	Y-axis offset to be applied to correction table 3 (bits) XML Example: <Tbl3YOff>0</Tbl3YOff>
Tbl3XGain	FLT	X-axis gain to be applied to correction table 3 XML Example: <Tbl3XGain>1.0</Tbl3XGain>

Table 15 - LENS CONFIGURATION DATA: HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
Tbl3YGain	FLT	Y-axis gain to be applied to correction table 3 XML Example: <Tbl3YGain>1.0</Tbl3YGain>
Tbl3Rotation	FLT	Field rotation to be applied to correction table 3 XML Example: <Tbl3Rotation>0.0</Tbl3Rotation>
Tbl4XOff	I32	X-axis offset to be applied to correction table 4 (bits) XML Example: <Tbl4XOff>0</Tbl4XOff>
Tbl4YOff	I32	Y-axis offset to be applied to correction table 4 (bits) XML Example: <Tbl4YOff>0</Tbl4YOff>
Tbl4XGain	FLT	X-axis gain to be applied to correction table 4 XML Example: <Tbl4XGain>1.0</Tbl4XGain>
Tbl4YGain	FLT	Y-axis gain to be applied to correction table 4 XML Example: <Tbl4YGain>1.0</Tbl4YGain>
Tbl4Rotation	FLT	Field rotation to be applied to correction table 4 XML Example: <Tbl4Rotation>0.0</Tbl4Rotation>
Data	N/A	End LensConfigData </Data>

6.3.5 CORRECTION TABLES

The correction table contains values to adjust laser location based on the lens distortion and laser galvo configuration.

Note: Correction table data may be changed by an application, but it is normally not. This data is usually provided by a marking head integrator using the characteristics of the lens and laser galvo configuration. Correction table data may also be sent to the SMC using the [sendStreamData](#) method. In this case, however, the data is not persistent and will be lost after session logout or reboot.

See also [requestFixedData](#) and [sendFixedData](#).

Correction Table Parametric Information

The following table contains the correction table parametric information, which is used for table design and manipulation.

Note: The following table contains groups of related parameters in contiguous rows. Each group begins with and ends with a bolded XML tag (e.g., **ReferenceInformation**).

Table 16 - CORRECTION TABLE PARAMETRIC INFORMATION

XML Tag	Type	Description/XML Example
Data	N/A	CorrTableData identifier <Data type='CorrTableData' rev='2.2'>
TableParams	N/A	Begin TableParams section <TableParams>
ReferenceInformation	N/A	Begin ReferenceInformation Section <ReferenceInformation>
Description	STR	Textual description of scan head configuration XML Example: <Description>ProSeries-1 14mm with Linos 163mm EFL lens</Description>
SourceScanHeadID	STR	Internal Cambridge Technology use XML Example: <SourceScanHeadID/>
SourceLensID	STR	Internal Cambridge Technology use XML Example: <SourceLensID/>
SourceSpacerID	STR	Internal Cambridge Technology use XML Example: <SourceSpacerID/>
TableRevision	STR	For customer reference XML Example: <TableRevision>A</TableRevision>

Table 16 - CORRECTION TABLE PARAMETRIC INFORMATION

XML Tag	Type	Description/XML Example
TableCreationDate	STR	For customer reference XML Example: <TableCreationDate> 12/31/2012 8:19 PM </TableCreationDate>
HeadType	STR	Internal Cambridge Technology use XML Example: <HeadType> LXP-10 </HeadType>
ReferenceInformation	N/A	End ReferenceInformation Section </ReferenceInformation>
Configuration	N/A	Begin Configuration Section <Configuration>
ThirdAxisPresent	BOOL	If true, this is a three-axis system with dynamic focus XML Example: <ThirdAxisPresent> false </ThirdAxisPresent>
PreserveCalFactors	BOOL	If true, adjust table contents to preserve CalFactor values in <u>mmToActuatorSpaceTransform</u> XML Example: <PreserveCalFactors> true </PreserveCalFactors>
CalibrateRectangularField	BOOL	If true, the field is calibrated as a rectangle XML Example: <CalibrateRectangularField> false </CalibrateRectangularField>

Table 16 - CORRECTION TABLE PARAMETRIC INFORMATION

XML Tag	Type	Description/XML Example
TableDataHasBeenCorrectedFromDesign	BOOL	If true the table contents reflect measurement-based iterations, not just theoretical content. XML Example: <TableDataHasBeenCorrectedFromDesign> true </TableDataHasBeenCorrectedFromDesign>
ActuatorUnits	STR	String enumeration representing the galvo (actuator) command units: bits-16: -32768 to 32767 (EC1000 backwards compatibility) bits-20: -524288 to 524287 (EC1000 20-bit enhanced mode backwards compatibility) bits-24: -8388608 to 8388607 (SMC Standard) field-fraction: -0.5 to 0.5 (Cambridge Technology UAPI for the SC500) radians: galvo mechanical angle in radians (ScanPack direct) XML Example: <ActuatorUnits> bits-24 </ActuatorUnits>
RTCCCompatibleFormat	BOOL	Table data is organized to support RTC Compatibility mode of the SMC. XML Example: <RTCCCompatibleFormat> false </RTCCCompatibleFormat>
Configuration	N/A	End Configuration Section </Configuration>

Table 16 - CORRECTION TABLE PARAMETRIC INFORMATION

XML Tag	Type	Description/XML Example
DesignErrorComponents	N/A	Begin DesignErrorComponents Section <DesignErrorComponents>
Mirrors	BOOL	If true, mirror (pincushion) error compensation is/was calculated and inserted into the table. XML Example: <Mirrors>true</Mirrors>
Lens	BOOL	If true, lens distortion error compensation is/was calculated and inserted into the table XML Example: <Lens>true</Lens>
DistortionFactor	FLT	Strength of lens distortion theoretical calculation XML Example: <DistortionFactor>-2.0</DistortionFactor>
PincushionFactor	FLT	Strength of pincushion theoretical calculation XML Example: <PincushionFactor>1.0</PincushionFactor>
DesignErrorComponents	N/A	End DesignErrorComponents Section </DesignErrorComponents>
HeadParameters	N/A	Begin <i>HeadParameters</i> Section. <HeadParameters>

Table 16 - CORRECTION TABLE PARAMETRIC INFORMATION

XML Tag	Type	Description/XML Example
XGalvoMechHalfAngle-deg	FLT	X-axis mechanical half-angle XML Example: <XGalvoMechHalfAngle-deg> 11.0 </XGalvoMechHalfAngle-deg>
YGalvoMechHalfAngle-deg	FLT	Y-axis mechanical half-angle XML Example: <YGalvoMechHalfAngle-deg> 11.0 </YGalvoMechHalfAngle-deg>
XtoYMirrorDist-mm	FLT	X - Y mirror face-to-face spacing XML Example: <XtoYMirrorDist-mm> 35.0 </XtoYMirrorDist-mm>
YMirrorToRefSurfaceDist-mm	FLT	Y mirror to bottom of the head reference surface distance XML Example: <YMirrorToRefSurfaceDist-mm> 75.0 </YMirrorToRefSurfaceDist-mm>
RefSurfaceToWorkSurfaceDist-mm	FLT	Bottom of the head reference surface to work surface distance XML Example: <RefSurfaceToWorkSurfaceDist-mm> 192.0 </RefSurfaceToWorkSurfaceDist-mm>
LensFocalLength-mm	FLT	Design focal length of the F-Theta lens if used XML Example: <LensFocalLength-mm> 163.0 </LensFocalLength-mm>

Table 16 - CORRECTION TABLE PARAMETRIC INFORMATION

XML Tag	Type	Description/XML Example
LensMaxMechHalfAngle-deg	FLT	Maximum mechanical half-angle of lens entrance pupil. A lens property. XML Example: <LensMaxMechHalfAngle-deg> 15.0 </LensMaxMechHalfAngle-deg>
XMirrorToObjectiveDist-mm	FLT	Distance from X mirror center to objective in three-axis systems XML Example: <XMirrorToObjectiveDist-mm> 225.0 </XMirrorToObjectiveDist-mm>
E1E2Spacing	FLT	Nominal distance between Objective and DFM lens for the design working distance/ field-size XML Example: <E1E2Spacing> 65.0 </E1E2Spacing>
ZCalFactorCoeffs	N/A	Begin ZCalFactorCoeffs Section. Z Calibration Factor Coefficients used to calculate the ZCal Factor as a function of <u>RefSurfaceToWorkSurfaceDist-mm</u> <ZCalFactorCoeffs>
An	FLT	As many entries as required to properly model the cal factor XML Example: <An>2456.34</An> XML Example: <An>-21.60</An>
ZCalFactorCoeffs	N/A	End ZCalFactorCoeffs Section. </ZCalFactorCoeffs>

Table 16 - CORRECTION TABLE PARAMETRIC INFORMATION

XML Tag	Type	Description/XML Example
e1e2Coeffs	N/A	Begin <i>e1e2Coeffs</i> Section. e1e2 spacing coefficients used to calculate the objective to DFM lens spacing as a function of <u>RefSurfaceToWorkSurfaceDist-mm</u> . <e1e2Coeffs>
An	FLT	As many entries as required to properly model the e1e2 spacing XML Example: <An>101.306</An> XML Example: <An>-.4079</An>
e1e2Coeffs	N/A	End <i>e1e2Coeffs</i> Section. </e1e2Coeffs>
HeadParameters	N/A	End HeadParameters Section. </HeadParameters>
mmToActuatorSpaceTransform	N/A	Begin mmToActuatorSpaceTransform Section. The data in the section represents a 3-axis transform that can be used to convert mm units into galvo command units. <mmToActuatorSpaceTransform>
Xx	FLT	Represents the conversion factor for the X-Axis in units of actuator-units per millimeter. XML Example: <Xx>551.48</Xx>
Yx	FLT	(Reserved for future use) XML Example: <Yx>0</Yx>

Table 16 - CORRECTION TABLE PARAMETRIC INFORMATION

XML Tag	Type	Description/XML Example
Zx	FLT	(Reserved for future use) XML Example: <Zx>0</Zx>
Dx	FLT	(Reserved for future use) XML Example: <Dx>0</Dx>
Xy	FLT	(Reserved for future use) XML Example: <Xy>0</Xy>
Yy	FLT	Represents the conversion factor for the Y-Axis in units of actuator-units per millimeter. XML Example: <Yy>551.48</Yy>
Zy	FLT	(Reserved for future use) XML Example: <Zy>0</Zy>
Dy	FLT	(Reserved for future use) XML Example: <Dy>0</Dy>
Xz	FLT	(Reserved for future use) XML Example: <Xz>0</Xz>
Yz	FLT	(Reserved for future use) XML Example: <Yz>0</Yz>
Zz	FLT	Represents the conversion factor for the Z-Axis in units of actuator-units per millimeter. XML Example: <Zz>1100</Zz>
Dz	FLT	(Reserved for future use) XML Example: <Dz>0</Dz>

Table 16 - CORRECTION TABLE PARAMETRIC INFORMATION

XML Tag	Type	Description/XML Example
mmToActuatorSpaceTransform	N/A	End mmToActuatorSpaceTransform section. </mmToActuatorSpaceTransform>
TableStructure	N/A	Begin TableStructure Section. Defines how to interpret the table data XML Example: <TableStructure>
XActuatorMin	FLT	Minimum ideal X actuator table value XML Example: <XActuatorMin>-32768</XActuatorMin>
XActuatorStride	FLT	Spacing between X actuator ideal values XML Example: <XActuatorStride>1024</XActuatorStride>
X-NumCols	I32	Number of columns for the X-axis XML Example: <X-NumCols>65</X-NumCols>
YActuatorMin	FLT	Minimum ideal Y actuator table value XML Example: <YActuatorMin>-32768</YActuatorMin>
YActuatorStride	FLT	Spacing between Y actuator ideal values XML Example: <YActuatorStride>1024</YActuatorStride>

Table 16 - CORRECTION TABLE PARAMETRIC INFORMATION

XML Tag	Type	Description/XML Example
Y-NumRows	I32	Number of rows for the Y-axis XML Example: <Y-NumRows>65</Y-NumRows>
ZActuatorMin	FLT	Minimum ideal Z actuator table value XML Example: <ZActuatorMin>-32768</ZActuatorMin>
ZActuatorStride	FLT	Spacing between Z actuator ideal values XML Example: <ZActuatorStride>1024</ZActuatorStride>
Z-NumLayers	I32	Number of layers for the Z-axis XML Example: <Z-NumLayers>1</Z-NumLayers>
TableStructure	N/A	End TableStructure Section </TableStructure>
TableParams	N/A	End <i>TableParams</i> section </TableParams>

Correction Table Hardware Initialization Settings

The following tables contain the actual correction values for the Correction Table. At run-time, the ideal command value is used to index the table and each value is added to the ideal command to create a “corrected” command which is delivered to the galvos. Bi-linear interpolation between the four closest table entries is used to compute corrections when the ideal command does not fall on a table entry.

Table 17 - CORRECTION TABLE HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
x-axis	FLT	<p>X-Axis Correction Data for Z=0 plane</p> <p>X-NumCols * Y-NumRows comma- or semi-colon-separated floating-point values in actuator-units defining the X-axis correction starting in the lowest negative coordinate (lower left Cartesian quadrant) traversing X first, to the highest positive coordinate (upper right Cartesian quadrant).</p> <p>XML Example: <x-axis>203.01; 195.24; 161.05; ...; -174.56; -190.21</x-axis></p>
y-axis	FLT	<p>Y-Axis Correction Data for Z=0 plane</p> <p>X-NumCols * Y-NumRows comma- or semi-colon-separated floating-point values in actuator-units defining the Y-axis correction starting in the lowest negative coordinate (lower left Cartesian quadrant) traversing X first, to the highest positive coordinate (upper right Cartesian quadrant).</p> <p>XML Example: <y-axis>337.98; 323.63; 288.23; ...; -288.98; -323.04</y-axis></p>
z-axis	FLT	<p>Z-Axis Correction Data for Z=0 plane</p> <p>X-NumCols * Y-NumRows comma- or semi-colon-separated floating-point values in actuator-units defining the Z-axis correction starting in the lowest negative coordinate (lower left Cartesian quadrant) traversing X first, to the highest positive coordinate (upper right Cartesian quadrant).</p> <p>XML Example: <z-axis>2.13; 2.14; 1.08; ...; 4.67; 5.32</z-axis></p>
Supplemental Layers	N/A	<p>Begin SupplementalLayers section.</p> <p>Additional Correction table layers in support of full 3D correction table usage (ScanPack)</p> <p>XML Example: <SupplementalLayers></p>
Layer	N/A	<p>Begin <i>Layer</i> section.</p> <p>Correction table layer data at a specific ZOffset in mm from the Z=0 plane. A positive ZOffset is above the Z-0 plane.</p> <p>XML Example: <Layer ZOffset='10.0'></p>

Table 17 - CORRECTION TABLE HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
x-axis	FLT	<p>X-Axis Correction Data for Z=0 plane</p> <p>X-NumCols * Y-NumRows comma- or semi-colon-separated floating-point values in actuator-units defining the X-axis correction starting in the lowest negative coordinate (lower left Cartesian quadrant) to the highest positive coordinate (upper right Cartesian quadrant).</p> <p>XML Example: <x-axis>203.01; 195.24; 161.05; ...; -174.56; -190.21</x-axis></p>
y-axis	FLT	<p>Y-Axis Correction Data for Z=0 plane</p> <p>X-NumCols * Y-NumRows comma- or semi-colon-separated floating-point values in actuator-units defining the Y-axis correction starting in the lowest negative coordinate (lower left Cartesian quadrant) to the highest positive coordinate (upper right Cartesian quadrant).</p> <p>XML Example: <y-axis>337.98; 323.63; 288.23; ...; -288.98; -323.04</y-axis></p>
z-axis	FLT	<p>Z-Axis Correction Data for Z=0 plane</p> <p>X-NumCols * Y-NumRows comma- or semi-colon-separated floating-point values in actuator-units defining the Z-axis correction starting in the lowest negative coordinate (lower left Cartesian quadrant) to the highest positive coordinate (upper right Cartesian quadrant).</p> <p>XML Example: <z-axis>2.13; 2.14; 1.08; ... ; 4.67; 5.32</z-axis></p>
Layer	N/A	<p>End <i>Layer</i> Section</p> <p>XML Example: </Layer></p>
Supplemental Layers	N/A	<p>End SupplementalLayers Section</p> <p>XML Example: </SupplementalLayers></p>
Data	N/A	<p>End CorrTable Data</p> <p>XML Example: </Data></p>

6.3.6 USER CONFIGURATION

The data in the User Configuration file is used by the marking application as needed. (The values in the User Configuration file are completely under the control of a marking application.)

Note: The offset, gain and rotation variables are independent of, and additive to, the equivalent lens correction table adjustment factor defined in the Lens Configuration file.

Note: The general purpose user variables can be used to store any information that a marking application wishes to make persistent across reboots of the controller. It is up to the application to interpret the UserVar data as required.

User Configuration Data: Header and Host Application Initialization

The following table contains the header and host application initialization settings of the User Configuration file.

Note: In the User Configuration file, the host application initialization settings are optional and are not used by the hardware.

Table 18 - USER CONFIGURATION DATA SETTINGS: HEADER AND HOST APPLICATION INITIALIZATION

XML Tag	Type	Description/XML Example
Data	N/A	UserConfigData identifier <Data type='UserConfigData' rev='1.0'>
UserVar1	ANY	General purpose user variable XML Example: <UserVar1>ABC</UserVar1>
UserVar2	ANY	General purpose user variable XML Example: <UserVar2>123</UserVar2>
UserVar3	ANY	General purpose user variable XML Example: <UserVar3>4.56</UserVar3>
UserVar4	ANY	General purpose user variable XML Example: <UserVar4>true</UserVar4>
UserVar5	ANY	General purpose user variable XML Example: <UserVar5>>false</UserVar5>

Table 18 - USER CONFIGURATION DATA SETTINGS: HEADER AND HOST APPLICATION INITIALIZATION

XML Tag	Type	Description/XML Example
UserVar6	ANY	General purpose user variable XML Example: <UserVar6>'text'</UserVar6>

User Configuration Data: Hardware Initialization Settings

The following table contains the hardware initialization settings for the User Configuration file.

Note: In the User Configuration file, the hardware initialization settings are required.

Table 19 - USER CONFIGURATION DATA: HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
XOff	FLT	Offset to be applied to all X-Axis coordinates (bits, or mm if specified in fractional format) XML Example: <XOff>0</XOff>
YOff	FLT	Offset to be applied to all Y-Axis coordinates (bits, or mm if specified in fractional format) XML Example: <YOff>0</YOff>
ZOff	FLT	Offset to be applied to all Z-Axis coordinates (bits, or mm if specified in fractional format) XML Example: <ZOff>0</ZOff>
XGain	FLT	Gain factor to be applied to all X-axis coordinates XML Example: <XGain>1.0</XGain>
YGain	FLT	Gain factor to be applied to all Y-axis coordinates XML Example: <YGain>1.0</YGain>
Rotation	FLT	Rotation transformation to be applied to the X-Y field XML Example: <Rotation>90.0</Rotation>
Data		End UserConfigData

Table 19 - USER CONFIGURATION DATA: HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
		</Data>

6.3.7 PERFORMANCE ADJUSTMENTS

The Performance Adjustments file contains values that are used to adjust job parameters while the job is executing. This is of particular value when jobs are stored locally and adjustments need to be made to compensate for laser degradation on a particular machine.

Note: The data in the User Configuration file is intended to be used by the marking application as needed.

See also requestFixedData and sendFixedData.

Performance Adjustments Data Header

The following table contains the Performance Adjustments file header.

Table 20 - PERFORMANCE ADJUSTMENTS DATA HEADER

XML Tag	Type	Description/XML Example
Data	N/A	PerformanceMatrixData identifier <Data type='PerformanceMatrixData' rev='2.0'>

Performance Adjustments Data: Hardware Initialization Settings

The following table contains the hardware initialization settings for the Performance Adjustments table.

Table 21 - PERFORMANCE ADJUSTMENTS DATA: HARDWARE INITIALIZATION SETTINGS

XML Tag	Type	Description/XML Example
LaserPower	FLT	Scale factor to be applied to the laser power value specified in the job XML Example: <LaserPower>1.0</LaserPower>
PulseWidth	FLT	Scale factor to be applied to the laser pulse width specified in the job XML Example: <PulseWidth>1.0</PulseWidth>
Period	FLT	Scale factor to be applied to the laser pulse period specified in the job XML Example: <Period>1.0</Period>
<u>MarkSpeed</u>	FLT	Scale factor to be applied to the <u>MarkSpeed</u> specified in the job XML Example: <MarkSpeed>1.0</MarkSpeed>
XOffset	I16	Offset to be applied to all X coordinates (bits, or mm if specified in fractional format) XML Example: <XOffset>0</XOffset>
YOffset	I16	Offset to be applied to all Y coordinates (bits, or mm if specified in fractional format) XML Example: <YOffset>0</YOffset>
ZOffset	I16	Offset to be applied to all Z coordinates (bits, or mm if specified in fractional format) XML Example: <ZOffset>0</ZOffset>
Data		End PerformanceMatrixData </Data>

6.3.8 SERVO CONFIGURATION

The Servo Config file contains information about the tuning of the servo/galvo system. This file is automatically generated for Lightning II galvo systems using the Firmware Loader utility. The

parameters are used by the SMC embedded ScanPack algorithms to model the dynamic behavior of an attached scan-head.



The data in the Servo Config file is critical to proper performance of the SMC and should not be changed unless under the guidance of Cambridge Technology Applications Engineering.

See also [requestFixedData](#) and [sendFixedData](#).

Servo Config Data

The following table contains the hardware initialization settings for the Servo Config file. There are separate sections for each axis of a multi-axis system. This table describes entries for a single axis.

Table 22 - SERVO CONFIG DATA

XML Tag	Type	Description/XML Example
Data	N/A	ServoConfigData identifier <Data type='ServoConfigData' rev='2.0'>
System_Dynamics	N/A	System_Dynamics section identifier <System_Dynamics>
SERVO_PARAMS	N/A	Axis specific SERVO_PARAMS section identifier <SERVO_PARAMS>
AxisID	STR	Axis identifier. Options are "X", "Y", and "Z" XML Example: <AxisID>X</AxisID>
CommandGain	FLT	Command gain of the servo system in radians/half-field XML Example: <CommandGain>-0.174</CommandGain>
BandWidthHz	FLT	The servo/galvo bandwidth in Hz XML Example: <BandWidthHz>4500.0</BandWidthHz>
Damping	FLT	Damping factor XML Example: <Damping>0.8</Damping>

Table 22 - SERVO CONFIG DATA

XML Tag	Type	Description/XML Example
IntegratorBandWidthHz	FLT	Error integrator bandwidth in Hz XML Example: <IntegratorBandWidthHz>0.0</IntegratorBandWidthHz>
PositionFF	FLT	Position loop feed-forward factor XML Example: <PositionFF>1.0</PositionFF>
VelocityFF	FLT	Velocity loop feed-forward factor XML Example: <VelocityFF>0.4</VelocityFF>
AccelFF	FLT	Acceleration loop feed-forward factor XML Example: <AccelFF>0.0</AccelFF>
FilterTimeSec	FLT	Roll-off filter time constant in seconds XML Example: <FilterTimeSec>8.534e-5</FilterTimeSec>
MaxVelocity	FLT	Maximum velocity in radians/sec XML Example: <MaxVelocity>220000.0</MaxVelocity>
MaxAccel	FLT	Maximum acceleration in radians/sec/sec XML Example: <MaxAccel>147465.0</MaxAccel>
SERVO_PARAMS	N/A	End SERVO_PARAMS section </SERVO_PARAMS>
System_Dynamics	N/A	End System_Dynamics section </ System_Dynamics>
Data	N/A	End ServoConfigData </Data>

6.4 MARKING JOB SPECIFICATION

The primary interface for interacting with the controller is the sendStreamData method. This method streams data to the controller as fast as the network and buffering systems allow. Buffering is distributed between the host operating system, the SMC operating system, the SMC control software, and finally, the marking engine input FIFO.

sendStreamData is non-blocking in the sense that it returns as soon as the data is passed to the downstream communications system for transfer to the target SMC. Once this method returns, subsequent calls can be made to keep the data "pipeline" full with marking data. This technique ensures continuous marking operation without pauses.

Job data passed to the SMC remains in vector format until it reaches the real-time marking engine controller. Only then is it converted to time-domain command data and passed to the laser galvo controllers.

6.4.1 JOB DATA TYPES

The streaming data interface can send several types of data:

1. JobData (standard) – This is data that represents a marking job using the XML-based job definition language described in the next section. This job data is executed immediately in the same sequence as it is sent through the interface.
2. JobData (structured) – This is data that uses XML constructs to group related job instructions together into a segment that can be loaded to the board one time, and referred to multiple times via a separate sequence definition. A sequence definition construct permits the ordering of execution and iteration of pre-loaded segments.
3. CorrTableData – This data is in the same format as the correction table XML definition. Correction table data sent this way does not persist through an SMC power cycle.

6.4.2 JOB DATA DEFINITION

Job data contains both action commands that direct the marking engine to perform specific operations, and parametric data that affects how the SMC hardware behaves. Parameter commands do not cause any action, but modify the behavior of subsequent action commands. To minimize the number of XML identifier tags to express a job, the XML definition make use of two types of constructs. All action commands use specific XML tag names to identify the action, followed by a

comma-separated list of argument values. The *set* tag is used with an attribute *id* to identify the parameter followed by a comma- or semicolon-separated list of values.

In its simplest form, a streaming job packet is a well-formed XML document that is delimited with the tag “Data”. For example, an empty job would look like:

```
<Data type='JobData' rev='2.0'></Data>
```

A more useful example of a simple job that draws a box would look like:

XML Text	Description
<Data type='JobData' rev='2.0'>	Job data type declaration
<set id='JumpDelay'>150</set>	The parameter 'JumpDelay' is set to 150μsec.
<set id='MarkDelay'>150</set>	The parameter 'MarkDelay' is set to 150μsec.
<set id='PolyDelay'>50</set>	The parameter 'PolyDelay' is set to 150μsec.
<set id='LaserTiming'>50</set>	Set the laser time base tick to 50 20nsec periods (1μsec).
<set id='LaserOnDelay'>75</set>	The parameter 'LaserOnDelay' is set to 75 laser timing ticks.
<set id='LaserOffDelay'>100</set>	The parameter 'LaserOffDelay' is set to 100 laser timing ticks.
<set id='LaserPulse'>1; 10; 20</set>	Set the modulation of LASER_MOD1 to a pulse width of 10 laser timing ticks with a period of 20 laser timing ticks.
<Set id='JumpSpeed'>10; 30</Set>	The parameter 'JumpSpeed' is set to 30 bits per each 10μ sec update period.
<Set id='MarkSpeed'>10; 10</Set>	The parameter 'MarkSpeed' is set to 10 bits per each 10μ update period.
<JumpAbs>-5000; -5000</JumpAbs>	Move laser galvos to the absolute position -5000, -5000 with the laser off
<MarkAbs>-5000; 5000</MarkAbs>	Move laser galvos to the absolute position -5000, 5000 with the laser on.
<MarkAbs>5000; 5000</MarkAbs>	Move laser galvos to the absolute position 5000, 5000 with the laser on.

XML Text	Description
<code><MarkAbs>5000; -5000</MarkAbs></code>	Move laser galvos to the absolute position 5000, -5000 with the laser on.
<code><MarkAbs>-5000; -5000</MarkAbs></code>	Move laser galvos to the absolute position -5000, -5000 with the laser on.
<code></Data></code>	End job data

6.4.3 JOB TYPE SPECIFICATION

As shown in the following example, the job type is defined in the header section of the job XML, which precedes the job commands.

XML Text	Description
<code><Data type='JobData' rev='2.0'></code>	Standard Job Data type declaration, streaming or structured.
<code><Data type='CorrTableData' rev='2.2'></code>	Correction Table data See definitions in that section. A correction table may be sent as a packet, but it is not persistent through a SMC reboot.

6.5 JOB PARAMETERS AND COMMANDS

Jobs are made up of parameter definitions and action commands. Parameters are defined using the Set tag. Multiple values for parameters are expressed in a comma-separated list. Commands are represented by a keyword and one or more arguments in a list. Parameters and commands are grouped by function in the following sections.

6.5.1 USER UNITS CONVERSION

ActuatorUnits		
Description	Specifies the scanner bit resolution expectation of the job data. This information is used to scale coordinate values to the 24-bit native resolution required by the SMC	
Syntax	<set id='ActuatorUnits'>{STR unitsID}</set>	
Example	<set id='ActuatorUnits'>bits-16</set>	
Arguments	unitsID	Identifies a conversion ratio.
	Value range	bits-16 – coordinate values are assumed to be for a 16-bit coordinate system. This is the default for backward compatibility with the EC1000. bits-20 – coordinate values are assumed to be for a 20-bit coordinate system as used by the 20-bit version of the EC1000. bits-24 – coordinate values are assumed to be for a 24-bit coordinate system as used by the SMC.
Units		
Description	Specifies the units of the job coordinate data. It implicitly sets the conversion ratio used to map a motion-related command values from the user units to internally required “bits” units. This command uses the current cal factors of the SMC as defined by the correction table file that is loaded at boot time. These values may be over-ridden by using the commands <set id='XYCalFactor'> and <set id='ZCalFactor'>. This command affects the units of all motion commands that reference a coordinate or offset.	
Syntax	<set id='Units'>{U16 unitsID}</set>	
Example	<set id='Units'>2</set>	
Arguments	unitsID	Identifies a conversion ratio.

ActuatorUnits		
	Value range	<p>0 - bits (1:1); Note: This is the default value.</p> <p>1 - mm ($\text{UserUnits} * \text{CalFactor}$)</p> <p>2 - inch ($\text{UserUnits} * 25.4 * \text{CalFactor}$)</p> <p>3 - mils ($(\text{UserUnits}/1000) * 25.4 * \text{CalFactor}$)</p> <p>Note: In options 1, 2, and 3, UserUnits is the motion-related command value.</p>

XYCalFactor		
Description	Sets the X- and Y-axis calibration factor used in converting coordinate system units. Note that this command overrides the cal factors that are read by the API from the correction table file during a session login.	
Syntax	<code><set id='XYCalFactor'>{FLT multiplier}</set></code>	
Example	<code><set id='XYCalFactor'>500</set></code>	
Arguments	multiplier	Used as a bits/millimeter multiplier in converting user motion command units into the internally required “bits” units. The actual ratio is defined per the <code><set id='Units'></code> command.
	Value range	<p>The minimum value is 0.</p> <p>The maximum value is the practical limit of the hardware.</p>

ZCalFactor	
Description	Sets the Z-axis calibration factor used in converting coordinate system units. Note that this command overrides the cal factor that is read by the API from the correction table file during a session login.
Syntax	<code><set id='ZCalFactor'>{FLT multiplier}</set></code>
Example	<code><set id='ZCalFactor'>125</set></code>

ZCalFactor		
Arguments	multiplier	Used as a bits/millimeter multiplier in converting user motion command units into the internally required “bits” units. The actual ratio is defined per the <set id='Units'> command.
	Value range	The minimum value is 0. The maximum value is the practical limit of the hardware.

6.5.2 MOTION CONTROL PARAMETERS

CmdRangeCheckMode		
Description	Sets the behavior of the command range checking feature.	
Syntax	<set id='CmdRangeCheckMode'>{U16 enable; U16 port; U16 value}1; 5; 1</set>	
Example	<set id='CmdRangeCheckMode'>1; 5; 1</set>	
Arguments	enable	Enable/disable checking
	Value range	0 – Disables checking 1 – Enables checking
	port	Digital output port to manipulate
	Value range	See WriteDigital .
	value	Port value to set if out of range
	Value range	See WriteDigital .

JumpDelay	
Description	Sets the delay used at the end of a jump command.

JumpDelay		
Syntax	<set id='JumpDelay'>{U16 duration}</set>	
Example	<set id='JumpDelay'> 150 </set>	
Arguments	duration	The length of time to delay (in µsecs)
	Value range	0 - 32767

JumpSpeed (Two argument syntax)		
Description	Establishes the vector speed at which a jump is executed. The parameters are normally derived from an application speed setting expressed as mm/sec, bits/msec, or some other appropriate ratio.	
Syntax	<set id='JumpSpeed'>{U16 stepTime; FLT stepSize}</set>	
Example	<set id='JumpSpeed'> 10; 30 </set>	
Arguments	stepTime	The duration in µsecs between each micro-step. This is how often the galvo position command is updated with an incremental stepSize.
	Value range	10 - 65535
	stepSize	The distance traveled in bits for each micro-step. This value can be an integer or fractional number. When a fractional number is used, it is limited to 10-bit precision and the maximum step size is reduced to 64.0.
	Value range	Integer: 1 – 65535 bits Fractional: .001 - 64.0

JumpSpeed (Single argument syntax)	
Description	Establishes the vector speed at which a jump is executed.
Syntax	<set id='JumpSpeed'>{FLT speed}</set>

JumpSpeed (Single argument syntax)		
Example	<set id='JumpSpeed'> 10000 </set>	
Arguments	speed	Jump vector speed; the argument is interpreted as a floating-point vector speed in user-units/second. The update rate is specified in the <u>JumpStepTime</u> parameter.
	Value range	The minimum value is >0. The maximum value is the practical limit of the hardware.

JumpStepTime		
Description	Sets the update interval to be used in defining the jumping speed when the command <set id='JumpSpeed'> is invoked with a single argument.	
Syntax	<set id='JumpStepTime'>{U16 value}</set>	
Example	<set id='JumpStepTime'> 10 </set>	
Arguments	value	<u>JumpSpeed</u> update interval (in µsecs). The default value is 10.
	Value range	The minimum value is 10. The maximum value is the practical limit of the hardware.

MarkDelay		
Description	Sets the delay used at the end of a series of marks.	
Syntax	<set id='MarkDelay'>{U16 duration}</set>	
Example	<set id='MarkDelay'> 150 </set>	
Arguments	duration	Length of time to delay (in µsecs)
	Value range	0 - 32767

MarkSpeed (Two argument syntax)		
Description	Establishes the vector speed at which a mark is executed. The parameters are normally derived from an application speed setting expressed as mm/sec, bits/msec, or some other appropriate ratio.	
Syntax	<set id='MarkSpeed'>{U16 stepTime; FLT stepSize}</set>	
Example	<set id='MarkSpeed'> 10; 30 </set>	
Arguments	stepTime	The duration in μ secs between each micro-step. This is how often the galvo position command is updated with an incremental stepSize.
	Value range	10 - 65535
	stepSize	The distance traveled in bits for each micro-step. This value can be an integer or fractional number. When a fractional number is used, it is limited to 10-bit precision and the maximum step size is reduced to 64.0.
	Value range	Integer: 1 – 65535 bits Fractional: .001 - 64.0

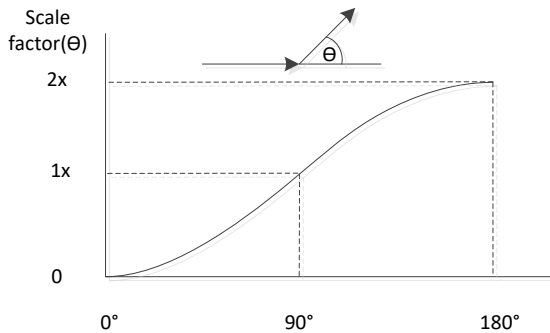
MarkSpeed (Single argument syntax)		
Description	Establishes the vector speed at which a mark is executed.	
Syntax	<set id='MarkSpeed'>{FLT speed}</set>	
Example	<set id='MarkSpeed'> 5000.0 </set>	
Arguments	speed	Mark vector speed; the argument is interpreted as a floating-point vector speed in user-units/second. The update rate is specified in the <u>MarkStepTime</u> parameter.
	Value range	The minimum value is >0. The maximum value is the practical limit of the hardware.

MarkStepTime		
Description	Sets the update interval to be used in defining the marking speed when the command <code><set id='MarkSpeed'></code> is invoked with a single argument.	
Syntax	<code><set id='MarkStepTime'>{U16 value}</set></code>	
Example	<code><set id='MarkStepTime'>10</set></code>	
Arguments	value	<u>MarkSpeed</u> update interval (in μ secs). The default value is 10.
	Value range	The minimum value is 10. The maximum value is the practical limit of the hardware.

PolyDelay		
Description	Set the delay to be used at the junction of two marks.	
Syntax	<code><set id='PolyDelay'>{U16 duration}</set></code>	
Example	<code><set id='PolyDelay'>150</set></code>	
Arguments	duration	The length of time (in μ secs) to delay between two sequential mark vectors
	Value range	0 - 32767

VariJumpDelay		
Description	Below a given jumpLengthLimit, the jump delay is linearly scaled down from the <u>JumpDelay</u> value to the minimumDelay as the jump distance approaches zero.	
Syntax	<code><set id='VariJumpDelay'>{U16 minimumDelay; U16 jumpLengthLimit}</set></code>	
Example	<code><set id='VariJumpDelay'>50; 2000</set></code>	
Arguments	minimumDelay	Minimum length (in laser timing ticks) of a jump delay
	Value range	0 - 65535

VariJumpDelay		
	jumpLengthLimit	Jump length threshold (in 1-bit user units) below which the variable jump delay will be applied
	Value range	1 - 65535

VariPolyDelayFlag		
Description	Enables or disables the use of variable polygon delays. If variable polygon delays are enabled, the <i>PolyDelay</i> value is adjusted proportional to the angular change in the next segment of the poly-vector.	
Syntax	<set id='VariPolyDelayFlag'>{ <i>BOOL value</i> }	
Example	<set id='VariPolyDelayFlag'>true</set>	
Arguments	value	Variable polygon delay enabled state
	Value range	true (enabled) false (disabled)
Comments	<p>When this feature is enabled, the <i>PolyDelay</i> value is scaled proportionate to the angle of the vertex. The scaling is according to a raised cosine function as shown below:</p> 	

Wobble		
Description	Allows varying line width during a Mark command. The marking vector is modified with a circular repeating pattern defined by the param and amplitude arguments.	
Syntax	<set id='Wobble'>{U16 param; FLT amplitude}</set>	
Example	<set id='Wobble'> 250; 10 </set>	
Arguments	param	Interpretation varies per the setting of the WobbleMode command. If WobbleMode = 1, this value represent the overlap of the wobble movement (in percent) If WobbleMode = 2, this value is the Period of the wobble movement (in μ secs)
	Value range	-500 – 99 (WobbleMode = 1) 20 – 65535 (WobbleMode = 2)
	amplitude	Amplitude of the circular wobble movement (in job units)
	Value range	Depends on job units

WobbleMode	
Description	Sets the mode of the Wobble command that allows varying line width during a Mark command. The mode sets how the parameters of the Wobble command are interpreted.
Syntax	<set id='WobbleMode'>{U16 mode}</set>
Example	<set id='WobbleMode'> 2 </set>

WobbleMode		
Arguments	mode	<p>Sets the mode of the wobble command.</p> <p>0 – Constant fluence (Reserved). The linear mark speed is adjusted such that the tangential velocity of the wobble pattern is performed at the mark speed while maintaining a specified overlap.</p> <p>1 – Constant linear mark speed with a specified overlap</p> <p>2 – Constant linear mark speed with a constant period (Default)</p>
	Value range	0 - 2

WobbleTable	
Description	<p>Allows a custom specification of the wobble pattern. Independent data can be specified for the X and Y axes for a table length of 1024. The table represents the values of a trajectory for the X and Y axes that will be repeated at the frequency specified in the <i>XFrequency</i> and <i>YFrequency</i> attributes. The table data is additive to the normal trajectory being used for marking.</p>
Syntax	<pre><WobbleTable XFrequency='{FLT XFreq}' YFrequency='{FLT YFreq}'> <Pt>{FLT XWobVal[0]; FLT YWobVal[0]}</Pt> <Pt>{FLT XWobVal[1]; FLT YWobVal[1]}</Pt> ... <Pt>{FLT XWobVal[1023]; FLT YWobVal[1023]}</Pt> </WobbleTable></pre>
Example	<pre><WobbleTable XFrequency='2.5' YFrequency='0.9' > <Pt>0.0; 0.0</Pt> <Pt>0.1; 0.05</Pt> <Pt>0.2; 0.1</Pt> ... <Pt>0.0; 0.0</Pt> </set></pre>

WobbleTable		
Arguments	XFrequency	Specifies the frequency in KHz that the X axis table will be repeated as it is applied to the marking vector.
	Value range	0.1 – 10.0KHz
	YFrequency	Specifies the frequency in KHz that the Y axis table will be repeated as it is applied to the marking vector.
	Value range	0.1 – 10.0KHz
	Pt	Specifies the X & Y offsets that will be added to the marking vector as the table is being traversed. Because the tables for the X and Y axes can be repeated at different frequencies, the table data pairings are not necessarily applied at the same time.
	Value range	Depends on job units

LissajousWobble	
Description	<p>Sets the parameters of a wobble pattern that is in the shape of a lissajous curve. This is a time and position variable pattern that pseudo-randomizes the wobble path. The formulas used to calculate the instantaneous wobble values is:</p> $Xw(t) = XAmplitude * \sin(\text{ToRadiansPerSec}(XFrequencyInKHz)(t) + \text{ToRadians}(XPhaseInDegrees))$ $Yw(t) = YAmplitude * \sin(\text{ToRadiansPerSec}(YFrequencyInKHz)(t) + \text{ToRadians}(XPhaseInDegrees))$ <p>NOTE: Because of the limited bandwidth of the galvos, there will be a frequency limit above which the wobble pattern will become distorted.</p>
Syntax	<set id='LissajousWobble'>{FLT XAmplitude; FLT YAmplitude; FLT XFrequencyInKHz; FLT YFrequencyInKHz; FLT XPhaseInDegrees}</set>
Example	<set id='LissajousWobble'> 0.1; 0.5; 1.0; 2.5; 45.0 </set>

LissajousWobble		
Arguments	XAmplitude	Amplitude of the X axis wobble pattern
	Value range	Depends on job units.
	YAmplitude	Amplitude of the Y axis wobble pattern
	Value range	Depends on job units.
	XFrequencyInKHz	Frequency of the X axis pattern repetition
	Value range	0.1 – 10KHz
	YFrequencyInKHz	Frequency of the Y axis pattern repetition
	Value range	0.1 – 10KHz
	XPhaseInDegrees	Phase relationship of the Y axis relative of the X axis at the start of the waveform generation
	Value range	+/- 180 degrees

WobbleEnable		
Description	Enables or disables the wobble function. Note: Wobble parameters should have already been set using the <Set id='Wobble'> parameter.	
Syntax	<WobbleEnable>{U16 wobbleEnableSetting}, [U16 direction]</WobbleEnable>	
Example	<WobbleEnable> 0, 1 </WobbleEnable>	
Arguments	wobbleEnableSetting	Indicates whether wobble is to be enabled or disabled.
	Value range	0 – Disable wobble 1 – Enable wobble
	Direction	Optional argument indicating the directionality of the circular wobble

WobbleEnable		
	Value range	0 – CCW rotation (default) 1 – CW rotation

XY2ErrorCheckMode (deprecated)		
Description	Enables or disables XY2-100 status checking and the generation of exceptions when the status is not as expected. Replaced by GalvoErrorCheckMode .	
Syntax	<set id='XY2ErrorCheckMode'>{U16 enable; HEX U32 mask; HEX U32 value}</set>	
Arguments	enable	If enabled, the XY2-100 and XY2-100e status registers are continuously evaluated by the following method: bit-wise ANDing the “mask” argument with the XY2-100 status words to select the bits to be evaluated and then comparing the selected bits to the “value” argument Note: The mask and value arguments are described below. If the value does not equal the actual masked status register, an exception is generated and marking is immediately halted. Recovery requires sending an Abort priority message to reset the logic.
	Value range	0 – Disable XY2-100 status checking and the generation of exceptions when the status is not as expected. 1 – Enable XY2-100 status checking and the generation of exceptions when the status is not as expected.
	mask	The mask is split into a lower (bits[15..0]) and upper (bits[31..16]) value corresponding to the primary XY2-100 port and secondary XY2-100e port, respectively. Note: See the description of the enable argument (above) to understand the interpretation of this value.
	Value range	0 - 0xFFFFFFFF

XY2ErrorCheckMode (deprecated)		
	value	The mask is split into a lower (bits[15..0]) and upper (bits[31..16]) value corresponding to the primary XY2-100 port and secondary XY2-100e port, respectively. Note: See the description of the enable argument (above) to understand the interpretation of this value.
	Value range	0 - 0xFFFFFFFF

GalvoErrorCheckMode		
Description	Enables or disables Galvo status checking and the generation of exceptions when the status is not as expected. XY2-100 or GSBUS status checking is automatically chosen based on the presence of devices on the GSBUS. If no GSBUS device is detected, then XY2-100 is assumed.	
Syntax	<set id='GalvoErrorCheckMode'>{U16 enable; HEX U32 mask; HEX U32 value}</set>	
Example	<set id='GalvoErrorCheckMode'>1; 0x0000FFFF; 0x0000FDFD</set>	
Arguments	enable	<p>If enabled and no GSBUS devices are present, the XY2-100 and XY2-100e status registers are continuously evaluated. If GSBUS devices are present, then the GSBUS status word is continuously evaluate by the following method:</p> <p>bit-wise ANDing the “mask” argument with the galvo status words to select the bits to be evaluated and then comparing the selected bits to the “value” argument</p> <p>Note: The mask and value arguments are described below.</p> <p>If the value does not equal the actual masked status register, an exception is generated and marking is immediately halted. Recovery requires sending an <u>Abort</u> priority message to reset the logic.</p>

GalvoErrorCheckMode		
	Value range	<p>0 – Disable XY2-100 (GSBus) status checking and the generation of exceptions when the status is not as expected.</p> <p>1 – Enable XY2-100 (GSBUS) status checking and the generation of exceptions when the status is not as expected.</p>
	mask	<p>For XY2-100, the mask is split into a lower (bits[15..0]) and upper (bits[31..16]) value corresponding to the primary XY2-100 port and secondary XY2-100e port, respectively. For the GSBus, the status is divided into eight 4-bit fields, one field for each galvo axis. The least significant four bits corresponds to the first axis found.</p> <p>Note: See the description of the enable argument (above) to understand the interpretation of this value.</p>
	Value range	0 - 0xFFFFFFFF
	value	<p>For XY2-100, the mask is split into a lower (bits[15..0]) and upper (bits[31..16]) value corresponding to the primary XY2-100 port and secondary XY2-100e port, respectively. For the GSBus, the status is divided into eight 4-bit fields, one field for each axis. The least significant four bits corresponds to the first axis found.</p> <p>Note: See the description of the enable argument (above) to understand the interpretation of this value.</p>
	Value range	0 - 0xFFFFFFFF

XY2AxisDisable (deprecated)	
Description	Enables or disables axis motion on the two XY2-100 ports. Replaced by <u>GalvoAxisDisable</u> .
Syntax	<set id='XY2AxisDisable'>{U16 XY2-100eAxisMask; U16 XY2-100AxisMask}</set>

XY2AxisDisable (deprecated)		
Example	<set id='XY2AxisDisable'> 0; 1 </set>	
Arguments	XY2-100eAxisMask	A three bit field disabling the corresponding axis on the secondary XY2-100e port. Bits[2..0] correspond to axes Z, Y, X, respectively. All bits == 0 means enable all axes on the head.
	Value range	0 - 7
	XY2-100AxisMask	A three bit field disabling the corresponding axis on the primary XY2-100 port. Bits[2..0] correspond to axes Z, Y, X, respectively.
	Value range	0 - 7

GalvoAxisDisable		
Description	Enables or disables axis motion on the two three-axis ports.	
Syntax	<set id='GalvoAxisDisable'>{U16 Head2AxisMask; U16 Head1AxisMask}</set>	
Example	<set id='GalvoAxisDisable'> 0; 1 </set>	
Arguments	Head2AxisMask	A three bit field disabling the corresponding axis on the secondary XY2-100-2 port. Bits[2..0] correspond to axes Z, Y, X, respectively. All bits == 0 means enable all axes on the head. For GSBUS based heads, Bits[2..0] correspond to axes , respectively.
	Value range	0 - 7
	Head1AxisMask	A three bit field disabling the corresponding axis on the primary XY2-100 port. Bits[2..0] correspond to axes Z, Y, X, respectively. All bits == 0 means enable all axes on the head. For GSBUS based heads, Bits[2..0] correspond to axes 5, 4, 3, respectively.
	Value range	0 - 7

GSBusDisable		
Description	Disables or enables active GSBus command channel driving by the SMC	
Syntax	<GSBusDisable>{bool disable}</GSBusDisable>	
Example	<GSBusDisable> true </GSBusDisable>	
Arguments	disable	Boolean indicating the desired state of GSBus command channel driving. Active command channel driving should be disabled if TuneMaster-II is used to examine and change tuning parameters of an attached Lightning-II scanner system.
	Value range	True or 1 – Disable driving the GSBus command channels False or 0 – Enable GSBus command channel driving

XY2AddressMode		
Description	Sets the addressing mode of the XY2-100 interface	
Syntax	<set id='XY2AddressMode'>{Mode}</set>	
Example	<set id='XY2AddressMode'> Normal </set>	
Arguments	Mode	Defines the command data width of the XY2-100 interface. <i>Normal</i> is traditional 16-bit command data. <i>Enhanced</i> is 20-bit command data used with Cambridge Technology Lightning™ II galvos with an XY2-100 interface.
	Value range	Normal or 0 Enhanced or 1

6.5.3 MOTION CONTROL COMMANDS

Note: Coordinate units are controlled by the <set id='Units'> parameter.

ArcAbs		
Description	Mark and Arc shape using the current marking parameters.	
Syntax	<ArcAbs>{FLT xCenter; FLT yCenter; FLT sweepAngle}</ArcAbs>	
Example	<ArcAbs> 1000; 2000; 47.5 </ArcAbs>	
Arguments	xCenter yCenter	Center of the arc
	Value range	$-2^{23} - (2^{23}-1)$ (bits) or +/- ½ field size (user units)
	sweepAngle	How far to sweep the arc in degrees. A positive value is counter-clockwise. The starting point of the arc is defined by the target of the last Jump or Mark instruction.
	Value range	0 - 360

EnableParkPosition (deprecated)		
Description	Enables or disables the “parking” of a scanhead in dual-scanhead systems. <u>GalvoAxisDisable</u> is the preferred command. Note: It is expected that a <u>JumpAbs</u> command is executed prior to this command to move the galvos to the “park” position.	
Syntax	<EnableParkPosition>{U16 parkSetting}</EnableParkPosition>	
Example	<EnableParkPosition> 2 </EnableParkPosition>	
Arguments	parkSetting	Identifies the scanhead(s) to be enabled or disabled

EnableParkPosition (deprecated)		
	Value range	0 - Parking is disabled for both heads 1 - Parking is enabled for head 1 (analog or XY2-extended port) 2 - Parking is enabled for head 2 (normal XY2-100 port) 3 - Both heads are parked.

JumpAbs		
Description	Moves laser galvos to the absolute position with the laser off.	
Syntax	<JumpAbs>{FLT xCoordinate; FLT yCoordinate[: FLT zCoordinate]}</JumpAbs>	
Example	<JumpAbs>-5000; -5000; 100</JumpAbs>	
Arguments	xCoordinate	X-coordinate of the end of a jump vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	-32768 – 32767 (bits) Note: Depending on the unit selection, this value can take on a range which represents the X field size of the system in floating point notation.
	yCoordinate	Y-coordinate of the end of a jump vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	-32768 – 32767 (bits) Note: Depending on the unit selection, this value can take on a range which represents the Y field size of the system in floating point notation.
	zCoordinate	Z coordinate of the end of a jump vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command. Note: The Z coordinate is optional. If the Z coordinate is not specified, the value of the Z coordinate is not changed.

JumpAbs		
	Value range	-32768 – 32767 (bits) Note: Depending on the unit selection, this value can take on a range which represents the Z field size of the system in floating point notation.

JumpAbsEx		
Description	Moves laser galvos to the absolute position with the laser off. Note: This command differs from the <u>JumpAbs</u> command (above) in that it permits values that exceed the 16-bit range of a normal scan head. This command is used in large virtual-field MOTF applications.	
Syntax	<JumpAbsEx>{FLT xCoordinate; FLT yCoordinate[; FLT zCoordinate]}</JumpAbsEx>	
Example	<JumpAbsEx>-50000; -65000; 1000</JumpAbsEx>	
Arguments	xCoordinate	X coordinate of the end of a jump vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the X virtual field size of the system in floating point notation.
	yCoordinate	Y coordinate of the end of a jump vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Y virtual field size of the system in floating point notation.

JumpAbsEx		
	zCoordinate	<p>Z coordinate of the end of a jump vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.</p> <p>Note: The Z coordinate is optional. If the Z coordinate is not specified, the value of the Z coordinate is not changed.</p>
	Value range	<p>$-2^{31} - 2^{31}-1$</p> <p>Note: Depending on the unit selection, this value can take on a range which represents the Z virtual field size of the system in floating point notation.</p>

JumpAbsList		
Description	Move the laser galvos to the each one of the specified points in succession at the specified update interval with the laser off.	
Syntax	<pre> <JumpAbsList tick='{U16 tick}'> <Pt>{FLT X0; FLT Y0; FLT Z0}</Pt> <Pt>{FLT X1; FLT Y1; FLT Z1}</Pt> ... <Pt>{FLT Xn; FLT Yn; FLT Zn}</Pt> </JumpAbsList> </pre>	
Example	<pre> <JumpAbsList tick='10'> <Pt>100; 215; 10</Pt> <Pt>110; 240; 30</Pt> <Pt>120; 250; 50</Pt> <Pt>130; 255; 60</Pt> </JumpAbsList> </pre>	
Arguments	tick	The galvo command update interval (in μsecs)
	Value range	10 – 65535

JumpAbsList		
	X_n	X coordinate in a sequence of point coordinates that will be written to the galvos at the rate specified by the tick parameter. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the X virtual field size of the system in floating point notation.
	Y_n	Y coordinate in a sequence of point coordinates that will be written to the galvos at the rate specified by the tick parameter. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Y virtual field size of the system in floating point notation.
	Z_n	Z coordinate in a sequence of point coordinates that will be written to the galvos at the rate specified by the tick parameter. Values are floating point and are converted into system “bits” units per the <u>Units</u> command. Note: The Z coordinate is optional. If the Z coordinate is not specified, the value of the Z coordinate is not changed.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Z virtual field size of the system in floating point notation.

Description	Moves the galvos to a position relative to the last commanded position with the laser off.
-------------	--

Syntax	<JumpRel>{FLT xOffset; FLT yOffset[; FLT zOffset]}</JumpRel>	
Example	<JumpRel> 25; 50; 0 </JumpRel>	
Arguments	xOffset	X offset used to calculate a jump vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	-65535 - 65535
	yOffset	Y offset used to calculate a jump vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	-65535 - 65535
	zOffset	Z offset used to calculate a jump vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the <u>Units</u> command. Note: The Z offset is optional. If the Z offset is not specified, its relative move is set to zero.
	Value range	-65535 - 65535

JumpRelEx	
Description	Moves the galvos to a position relative to the last commanded position with the laser off. Note: This command differs from the <u>JumpRel</u> command (above) in that it permits values that exceed the 16-bit range of a normal scan head. This command is used in large virtual-field MOTF applications.
Syntax	<JumpRelEx>>{FLT xOffset; FLT yOffset[; FLT zOffset]}</JumpRelEx>
Example	<JumpRelEx>> -50000; -65000; 1000 </JumpRelEx>

JumpRelEx		
Arguments	xOffset	X offset used to calculate a jump vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$
	yOffset	Y offset used to calculate a jump vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{32} - 2^{32}-1$
	zOffset	Z offset used to calculate a jump vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the <u>Units</u> command. Note: The Z coordinate is optional. If the Z coordinate is not specified, the value of the Z coordinate is not changed.
	Value range	$-2^{32} - 2^{32}-1$

JumpAndDrillList	
Description	Moves the galvos to each one of the specified points in succession and fires the laser. The laser properties are changed at each location as specified in the mode of operation. Note: Jumps are not micro-vectored, so galvo instability may result if jump distances are too large.
Syntax	<pre><JumpAndDrillList LaserOnTime='{U16 laserOnTime}' LaserFireMode='{Enum laserFireMode}' [ExtSyncPin ='{U16 extSyncPin }'] [ExtSyncPinState='{U16 extSyncPinState }']> <Pt>{FLT X0; FLT Y0}</Pt> <Pt>{FLT X1; FLT Y1}</Pt> ... <Pt>{FLT Xn; FLT Yn}</Pt></pre>

JumpAndDrillList		
	</JumpAndDrillList >	
Example	<pre>< JumpAndDrillList LaserOnTime='20' LaserFireMode='NoWait'> <Pt>100; 215</Pt> <Pt>110; 240</Pt> <Pt>120; 250</Pt> <Pt>130; 255</Pt> </JumpAndDrillList></pre>	
Argument	LaserOnTime	Specifies the duration that the laser is fired (in laser ticks).
	Value range	0 - 65535
	LaserFireMode	Specifies the synchronization level of issuance of the next jump relative to the firing of the laser.
	Value range	<p>NoWait – Fire the laser and do not wait. Immediately jump to the next location.</p> <p>WaitUntilOn – Fire the laser and wait until it actually is on. This accommodates any <u>LaserOnDelay</u> that may be specified.</p> <p>WaitUntilOff – Fire the laser and wait until it is off. This accomodates the <u>LaserOnDelay</u>, <u>LaserOnTime</u>, and <u>LaserOffDelay</u></p> <p>WaitUntilExtSync – Fire the Laser and wait until an external signal specified by the optional argument <i>ExtSyncPin</i> is asserted to the state specified by the optional argument <i>ExtSyncPinState</i>.</p> <p>WaitUntilGalvoCmdDelayComp – Fire the laser and wait for the amount of time (<i>LaserOnTime</i> – <u>GalvoCmdDelayComp</u>). <u>GalvoCmdDelayComp</u> is specified using the command: <code><set id='GalvoCmdDelayComp'></code>. If the result is negative, then jump to the next site immediately.</p>
	ExtSyncPin	Optional argument required only if the <i>LaserFireMode</i> is set to <i>WaitUntilExtSync</i> . Specifies the external pin to sense. The pin identifier is the same as the portNumber argument in the command <u>WaitForIO</u>

JumpAndDrillList		
	Value range	0 – 31 See WaitForIO for details.
	ExtSyncPinState	Optional argument required only if the <i>LaserFireMode</i> is set to <i>WaitUntilExtSync</i> . Specifies the logical state of the external pin being sensed. The state should take into consideration assertion inversions due to signal conditioning circuitry.
	Value range	0 – 1
	X_n	X coordinate in a sequence of point coordinates that will be written to the galvos in succession. The position values are floating point and are converted into system “bits” units per the Units command.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the X virtual field size of the system in floating point notation.
	Y_n	Y coordinate in a sequence of point coordinates that will be written to the galvos in succession. The position values are floating point and are converted into system “bits” units per the Units command.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Y virtual field size of the system in floating point notation.

GalvoCmdDelayComp	
Description	Sets the value to use in the JumpAndDrillList command when the <i>LaserFiringMode</i> is set to <i>WaitUntilGalvoCmdDelayComp</i>
Syntax	<set id='GalvoCmdDelayComp'>{U16 value}</set>
Example	<set id='GalvoCmdDelayComp'>30</set>

GalvoCmdDelayComp		
Arguments	value	Number of micro-seconds that represent the delay from issuing a jump command to when the galvos indicate out of position
	Value range	0 – 500

JumpAndFireList		
Description	<p>Moves the galvos to each one of the specified points in succession and fires the laser. The laser properties are changed at each location as specified in the mode of operation.</p> <p>Note: Jumps are not micro-vectored, so galvo instability may result if jump distances are too large.</p>	
Syntax	<pre><JumpAndFireList LaserOnTime='{U16 LaserOnTime}' LaserOnDelay='{U16 LaserOnDelay}' [OutputMode='{U16 outputMode}']> <Pt>{FLT X0; FLT Y0; FLT Z0; U32 laserValue0}</Pt> <Pt>{FLT X1; FLT Y1; FLT Z1; U32 laserValue1}</Pt> ... <Pt>{FLT Xn; FLT Yn; FLT Zn; U32 laserValuen}</Pt> </JumpAndFireList></pre>	
Example	<pre>< JumpAndFireList LaserOnTime='10' LaserOnDelay='0' OutputMode='0'> <Pt>100; 215; 10; 1</Pt> <Pt>110; 240; 30; 2</Pt> <Pt>120; 250; 50; 3</Pt> <Pt>130; 255; 60; 0</Pt> </JumpAndFireList></pre>	
Arguments	LaserOnTime	Specifies the duration that the laser is fired (in laser ticks).
	Value range	0 - 65535
	LaserOnDelay	Specifies the waiting period (in µsecs) before firing after an incremental jump.
	Value range	0 - 65535
	outputMode	Specifies how to interpret <i>laserValuen</i> .

JumpAndFireList		
	Value range	<p>0 = Interpret <i>laserValuen</i> as a laser pulse-width pair (laser-ticks)</p> <p>1 = Interpret <i>laserValuen</i> as Analog Port 1 value (12-bits)</p> <p>2 = Interpret <i>laserValuen</i> as Analog Port 2 value (12-bits)</p> <p>3 = Interpret <i>laserValuen</i> as Digital power port value (8-bits)</p>
	X_n	<p>X coordinate in a sequence of point coordinates that will be written to the galvos in succession.</p> <p>The position values are floating point and are converted into system “bits” units per the <u>Units</u> command.</p>
	Value range	<p>$-2^{31} - 2^{31}-1$</p> <p>Note: Depending on the unit selection, this value can take on a range which represents the X virtual field size of the system in floating point notation.</p>
	Y_n	<p>Y coordinate in a sequence of point coordinates that will be written to the galvos in succession.</p> <p>The position values are floating point and are converted into system “bits” units per the <u>Units</u> command.</p>
	Value range	<p>$-2^{31} - 2^{31}-1$</p> <p>Note: Depending on the unit selection, this value can take on a range which represents the Y virtual field size of the system in floating point notation.</p>
	Z_n	<p>Z coordinate in a sequence of point coordinates that will be written to the galvos in succession.</p> <p>The position values are floating point and are converted into system “bits” units per the <u>Units</u> command.</p>

JumpAndFireList		
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Z virtual field size of the system in floating point notation.
	laserValuen	Value is interpreted per the mode specification. The value is applied to the hardware prior to the laser being fired.
	Value range	The value range is mode-dependent. If outputMode = 0, the value represents individual pulse width settings in laser ticks for LASER_MOD1 and LASER_MOD2. The LASER_MOD1 setting is specified in bits [15 – 0], and the LASER_MOD2 setting in bits [31 – 16]

MarkAbs		
Description	Moves the galvos to the absolute position with the laser on.	
Syntax	<MarkAbs>{FLT xCoordinate; FLT yCoordinate[; FLT zCoordinate]}/</MarkAbs>	
Example	<MarkAbs>-5000; 5000; 200</MarkAbs>	
Arguments	xCoordinate	X coordinate of the end of a marking vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-32768 - 32767$ (bits) Note: Depending on the unit selection, this value can take on a range which represents the X field size of the system in floating point notation.
	yCoordinate	Y coordinate of the end of a marking vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.

MarkAbs		
	Value range	-32768 – 32767 (bits) Note: Depending on the unit selection, this value can take on a range which represents the Y field size of the system in floating point notation.
	zCoordinate	Z coordinate of the end of a marking vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command. Note: The Z coordinate is optional. If the Z coordinate is not specified, the value of the Z coordinate is not changed.
	Value range	-32768 – 32767 (bits) Note: Depending on the unit selection, this value can take on a range which represents the Z field size of the system in floating point notation.

MarkAbsEx		
Description	Moves the galvos to the absolute position with the laser on. Note: This command differs from the <u>MarkAbs</u> command (above) in that it permits values that exceed the 16-bit range of a normal scan head. This command is used in large virtual-field MOTF applications.	
Syntax	<MarkAbsEx>{FLT xCoordinate; FLT yCoordinate; FLT zCoordinate}</MarkAbsEx>	
Example	<MarkAbsEx>-50000; -65000; 0</MarkAbsEx>	
Arguments	xCoordinate	X coordinate of the end of a marking vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the X virtual field size of the system in floating point notation.

MarkAbsEx		
	yCoordinate	Y coordinate of the end of a marking vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Y virtual field size of the system in floating point notation.
	zCoordinate	Z coordinate of the end of a marking vector. Values are floating point and are converted into system “bits” units per the <u>Units</u> command. Note: The Z coordinate is optional. If the Z coordinate is not specified, the value of the Z coordinate is not changed.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Z virtual field size of the system in floating point notation.

MarkAbsList	
Description	Move the galvos to each of the specified points in succession, at the specified update interval, with the laser on. A Mark delay is inserted at the end of the list.
Syntax	<pre> <MarkAbsList tick='{U16 tick}'> <Pt>{FLT X0; FLT Y0; FLT Z0}</Pt> <Pt>{FLT X1; FLT Y1; FLT Z1}</Pt> ... <Pt>{FLT Xn; FLT Yn; FLT Zn}</Pt> </MarkAbsList> </pre>

MarkAbsList		
Example	<pre> <MarkAbsList tick='10'> <Pt>100; 215; 10</Pt> <Pt>110; 240; 30</Pt> <Pt>120; 250; 50</Pt> <Pt>130; 255; 60</Pt> </MarkAbsList> </pre>	
Arguments	tick	The galvo command update interval (in μ secs)
	Value range	10 - 65535
	X_n	X coordinate in a sequence of point coordinates that will be written to the galvos at the rate specified by the tick parameter. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the X virtual field size of the system in floating point notation.
	Y_n	Y coordinate in a sequence of point coordinates that will be written to the galvos at the rate specified by the tick parameter. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Y virtual field size of the system in floating point notation.
	Z_n	Z coordinate in a sequence of point coordinates that will be written to the galvos at the rate specified by the tick parameter. Values are floating point and are converted into system “bits” units per the <u>Units</u> command. Note: The Z coordinate is optional. If the Z coordinate is not specified, the value of the Z coordinate is not changed.

MarkAbsList		
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Z virtual field size of the system in floating point notation.

MarkRel		
Description	Moves the galvos to a position relative to the last commanded position with the laser on.	
Syntax	<MarkRel>{FLT xOffset; FLT yOffset[; FLT zOffset]}</MarkRel>	
Example	<MarkRel>-355; 500; 10</MarkRel>	
Arguments	xOffset	X offset used to calculate a marking vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the Units command.
	Value range	-65535 - 65535
	yOffset	Y offset used to calculate a marking vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the Units command.
	Value range	-65535 - 65535
	zOffset	Z offset used to calculate a marking vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the Units command. Note: The Z offset is optional. If the Z offset is not specified, its relative move is set to zero.
	Value range	-65535 - 65535

MarkRelEx		
Description	<p>Moves the galvos to a position relative to the last commanded position with the laser off.</p> <p>Note: This command differs from the <u>MarkRel</u> command (above) in that it permits values that exceed the 16-bit range of a normal scan head for virtual field MOTF applications.</p>	
Syntax	<MarkRelEx>{FLT xOffset; FLT yOffset[; FLT zOffset]}</MarkRelEx>	
Example	<MarkRelEx>-355; 500; 100</MarkRelEx>	
Arguments	xOffset	X offset used to calculate a marking vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$
	yOffset	Y offset used to calculate a marking vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$
	zOffset	<p>Z offset used to calculate a marking vector relative to the last commanded position. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.</p> <p>Note: The Z coordinate is optional. If the Z coordinate is not specified, the value of the Z coordinate is not changed.</p>
	Value range	$-2^{31} - 2^{31}-1$

6.5.4 LASER CONTROL PARAMETERS

LaserEnableDelay		
Description	Sets the time required to enable the laser prior to marking. A default value for this parameter can be set in the Laser Configuration file as parameter <u>LaserEnableDelay</u> .	
Syntax	<set id='LaserEnableDelay'>{U16 delay}</set>	
Example	<set id='LaserEnableDelay'> 10 </set>	
Arguments	delay	The delay (in milliseconds) from the leading edge of LASERENABLE to the leading edge of LASER_GATE
	Value range	0 - 65535

LaserEnableTimeout		
Description	Sets the time-out for LASERENABLE to de-assert. A default value for this parameter can be set in the Laser Configuration file as parameter <u>LaserEnableTimeout</u> .	
Syntax	<set id='LaserEnableTimeout'>{U16 timeout}</set>	
Example	<set id='LaserEnableTimeout'> 20 </set>	
Arguments	timeout	The time-out (in milliseconds) from the trailing edge of LASER_GATE to when LASERENABLE is de-asserted
	Value range	0 - 65535

LaserFPK	
Description	Sets the LaserFPK signal timing. A Default values for this parameter can be set in the Laser Configuration file as parameter <u>LaserFPK</u> .
Syntax	<set id='LaserFPK'>{FLT position; I16 length}</set>
Example	<set id='LaserFPK'> -100; 10 </set>

LaserFPK		
Arguments	position	The delay (in laser timing ticks) from the leading edge of LASER_GATE to the assertion of the LASER_MOD3 signal
	Value range	-32768 - 32767
	length	The duration (in laser timing ticks) of assertion of the LASER_MOD3 signal
	Value range	0 - 65535

LaserModDelay		
Description	Sets the modulation delay of the laser. A default value for this parameter can be set in the Laser Configuration file as parameter <u>LaserModDelay</u> .	
Syntax	<set id='LaserModDelay'>{U16 delay}</set>	
Example	<set id='LaserModDelay'> 25 </set>	
Arguments	delay	The delay (in laser timing ticks) from the leading edge of LASER_GATE to the output of the first pulse on the LASER_MOD1 signal
	Value range	0 - 65535

LaserOffDelay		
Description	Sets the delay for turning off the laser when marking.	
Syntax	<set id='LaserOffDelay'>{U16 duration}</set>	
Example	<set id='LaserOffDelay'> 100 </set>	
Arguments	duration	Length of time (in µsecs) to delay
	Value range	0 - 65535

LaserOnDelay		
Description	Sets the delay for turning on the laser when marking relative to micro-vector generation. A negative value means that LASER_GATE is asserted before micro-vectoring begins.	
Syntax	<set id='LaserOnDelay'> <i>{I16 duration}</i> </set>	
Example	<set id='LaserOnDelay'> 200 </set>	
Arguments	duration	Length of time to delay (in μ secs) relative to the start of micro-vectoring
	Value range	-32768 - 32767

LaserStandby		
Description	Sets the standby settings of the laser. A default value for this parameter can be set in the Laser Configuration file as parameter <u>LaserStandby</u> .	
Syntax	<set id='LaserStandby'>{U16 laserID; U16 width; U16 period}</set>	
Example	<set id='LaserStandby'> 2; 10; 100 </set>	
Arguments	laserID	Laser modulation signal identification
	Value range	1 = LASER_MOD1 2 = LASER_MOD2
	width	The width of the laser modulation pulse (in laser timing ticks) when the laser is ON.
	Value range	0 – 65535 Note: If the value is 0, no modulation will be emitted.
	period	The period of the laser modulation pulse train (in laser timing ticks) when the laser is ON.
	Value range	0 - 65535 Note: If the value is 0, no modulation will be emitted.

LaserPipelineDelay		
Description	Set the time that all laser signals are time-shifted relative to the issuance of galvo position commands. This delay is useful for compensating for digital servo controllers that have an inherent processing delay time from when the command input is applied to when the mirrors actually move.	
Syntax	<set id='LaserPipelineDelay'>{U16 delay}</set>	
Example	<set id='LaserPipelineDelay'>1500</set>	
Arguments	delay	The length of time (in μ secs) that all laser control signals are time-shifted relative to micro-vectoring operations.
	Value range	0 – 4000 Note: The value range is limited by the value of a laser timing tick. The capacity of the pipeline is 4000 tick elements.

Description	Sets the level of the laser power port. (Note: The LaserPower port may be the 8-bit digital port or analog port 0 depending on the Laser Configuration file setting of the <u>Laser Power Port</u> bit of the LaserModeConfig word.)	
Syntax	<set id='LaserPower'>{U16 powerValue}</set>	
Example	<set id='LaserPower'>200</set>	
Arguments	powerValue	Setting of the laser power port (in bits). If the value is different from the one in the last LaserPower command, then the <u>LaserPowerDelay</u> delay is invoked.
	Value range	0 - 255

LaserPowerDelay	
Description	Sets the delay after changing the power setting. A default value can be set in the Laser Configuration file as the parameter <u>LaserPowerDelay</u> .
Syntax	<set id='LaserPowerDelay'>{U32 duration}</set>
Example	<set id='LaserPowerDelay'>125</set>

LaserPowerDelay		
Arguments	duration	Length of time to delay after setting <u>LaserPower</u> or executing <u>WriteAnalog</u> for port 0
	Value range	0 - $(2^{32}-1)/50$

LaserPulse		
Description	Sets the laser ON pulse settings of the laser.	
Syntax	<set id='LaserPulse'>{U16 laserID; U16 width; U16 period}</set>	
Example	<set id='LaserPulse'> 1; 50; 100 </set>	
Arguments	laserID	Laser modulation signal identification
	Value range	1 = LASER_MOD1 2 = LASER_MOD2
	width	The width of the laser modulation pulse (in laser timing ticks) when the laser is ON
	Value range	0 – 65535 laser ticks
	period	<p>If <i>laserID</i> is set to 1, this value controls the repetition interval of the laser modulation pulse train (in laser timing ticks) for both LASER_MOD1 and LASER_MOD2 when the laser is ON.</p> <p>If <i>laserID</i> is set to 2, then this value sets the timing skew, or delay between LASER_MOD1 and LASER_MOD2. If the value is zero, then the two signals are in phase with each other. If the value is set to ½ of the period value used when <i>laserID</i> is 1, then the signals will be 180 degrees out of phase.</p> <p>Note: For backward compatibility with the EC1000 behavior, the signals will be 180 degrees out of phase if the period value is set to the same number when <i>laserID</i> is 1 and 2.</p>

LaserPulse		
	Value range	0 – 65535 laser ticks Note: If <i>laserID</i> is set to 2, then this value should be < the value set when <i>laserID</i> is set to 1.

LaserTiming		
Description	Sets the value of a laser timing "tick," which is the unit of measurement for all laser timing values. A default value can be set in the Laser Configuration file as the parameter <u>LaserTiming</u> .	
Syntax	<set id='LaserTiming'>{U16 value}</set>	
Example	<set id='LaserTiming'>50</set>	
Arguments	value	Number of 20ns clock period increments in a laser timing "tick"
	Value range	1 – 500

LaserModType		
Description	Sets the behavior laser modulation synchronization feature. A default value can be set in the Laser Configuration file by setting the Laser Sync Mode bits in the parameter <u>LaserModeConfig</u> . Use of this command effectively changes the setting those bits at run-time.	
Syntax	<set id='LaserModType'>{U16 type}</set>	
Example	<set id='LaserModType'>1</set>	
Arguments	type	Laser synchronization method

LaserModType			
	Value range	0 =	Asynchronous modulation. The laser modulation is discontinuous, switching between the background modulation and the lasing modulation coincident with the LASER_GATE signal
		1 =	Synchronous to the free-running modulation signal on LASER_MOD3. LASER_MOD3 takes its modulation settings from the background settings for LASER_MOD1. The background signal for LASER_MOD1 and LASER_MOD2 is set for no modulation. In this mode, the LASER_GATE and subsequent LASER_MOD1 and LASER_MOD2 timing is synchronized to the rising edge of pulses on LASER_MOD3
		2 =	Synchronous to the free-running modulation of LASER_MOD2. In this mode the LASER_GATE signal is synchronized to the falling edge of LASER_MOD2. Both LASER_MOD1 and LASER_MOD2 are free-running according to the LaserPulse settings defined for them.
		3 =	Synchronous to the external signal source received on LASER_STAT6. In this mode, the LASER_GATE and subsequent LASER_MOD1 and LASER_MOD2 timing is synchronized to the falling edge of pulses received on LASER_STAT6.
		Note: Synchronous modulation is useful for lasers that phase-lock to an external frequency source.	

6.5.5 LASER CONTROL COMMANDS

EnableLaser	
Description	Sets the laser active state.
Syntax	<code><set id='EnableLaser'>{<i>BOOL laserActiveState</i>}</set></code>
Example	<code><set id='EnableLaser'>TRUE</set></code>

EnableLaser				
Arguments	laserActive State	The “enabled” or “disabled” state of the laser		
	Value range		true =	The laser is enabled.
			false =	The laser is disabled. (If this value is selected, then the special pointer laser mode is activated per the settings of LaserModeConfig.)

LaserOn		
Description	Turns the laser on for the specified duration.	
Syntax	<LaserOn>{U32 duration}</LaserOn>	
Example	<LaserOn> 1000 </LaserOn>	
Arguments	duration	Length of time (in µsecs) to turn the laser on
	Value range	1 - 2 ³² -1

LaserFire		
Description	Turns the laser on for the specified duration and then pauses according to the command processing wait mode.	
Syntax	<LaserFire>{U16 waitMode; U16 duration}</LaserFire>	
Example	<LaserFire> 2; 1000 </LaserFire>	
Arguments	waitMode	The command processing wait mode.

LaserFire			
	Value range		0 = Do not wait. Process the next command immediately while lasing.
			1 = Wait until the laser starts firing. If a <u>LaserOnDelay</u> value is set, command processing is suspended until after that time.
			2 = Wait until complete. Command processing is suspended until the laser turns off, including the <u>LaserOffDelay</u> .
	duration	Length of time (in µsecs) to turn the laser on	
	Value range	1 - 65535	

LaserSignalOff		
Description	Turns the laser off immediately.	
Syntax	<LaserSignalOff></LaserSignalOff>	
Example	<LaserSignalOff></LaserSignalOff>	
Arguments	N/A	N/A
	N/A	N/A

LaserSignalOn		
Description	Turns the laser on immediately.	
Syntax	<LaserSignalOn></LaserSignalOn>	
Example	<LaserSignalOn></LaserSignalOn>	
Arguments	N/A	N/A

LaserSignalOn		
	N/A	N/A

6.5.6 EXTERNAL I/O COMMANDS

WaitForIO																				
Description	Wait for the digital port value to be set. Job execution will pause until the external signal is in the condition specified by the levelPolarity argument.																			
Syntax	<WaitForIO>{U16 portNum; U16 levelPolarity; U32 timeOut; U16 debounce }</WaitForIO>																			
Example	<WaitForIO> 2; 1; 100000; 5000 </WaitForIO>																			
Arguments	portNumber	Port identifier																		
	Value range	<table><tr><td>Port</td><td>Association</td></tr><tr><td>0</td><td>AUX_START_ISO</td></tr><tr><td>1-4</td><td>AUX_GPI{1-4}_ISO</td></tr><tr><td>5</td><td>START</td></tr><tr><td>6</td><td>ABORT_ISO</td></tr><tr><td>7-13</td><td>LASER_STAT{0-6}_ISO</td></tr><tr><td>14</td><td>XY2_INPOS</td></tr><tr><td>15</td><td>AUX_XY2_INPOS</td></tr><tr><td>16-31</td><td>AUX_GPI{0-15}</td></tr></table>	Port	Association	0	AUX_START_ISO	1-4	AUX_GPI{1-4}_ISO	5	START	6	ABORT_ISO	7-13	LASER_STAT{0-6}_ISO	14	XY2_INPOS	15	AUX_XY2_INPOS	16-31	AUX_GPI{0-15}
	Port	Association																		
	0	AUX_START_ISO																		
1-4	AUX_GPI{1-4}_ISO																			
5	START																			
6	ABORT_ISO																			
7-13	LASER_STAT{0-6}_ISO																			
14	XY2_INPOS																			
15	AUX_XY2_INPOS																			
16-31	AUX_GPI{0-15}																			
	levelPolarity	Defines the state or transition of the signal when triggering occurs.																		
	Value range	<table><tr><td>0 = LowLevel</td></tr><tr><td>1 = HighLevel</td></tr><tr><td>2 = RisingEdge</td></tr><tr><td>3 = FallingEdge</td></tr></table>	0 = LowLevel	1 = HighLevel	2 = RisingEdge	3 = FallingEdge														
0 = LowLevel																				
1 = HighLevel																				
2 = RisingEdge																				
3 = FallingEdge																				

WaitForIO		
	timeOut	The wait is aborted if the time exceeds this value (in μ secs). If set to 0, the wait is indefinite. If a time-out occurs, an exception event is generated and the WaitForIOTimeout error code returned.
	Value range	0 - $(2^{32}-1)/50$
	debounce	Length of time (in milliseconds) to debounce the external signal
	Value range	1 - 65535

WriteAnalog		
Description	Sets the analog output port to a new value.	
Syntax	<WriteAnalog>{U16 PortNumber; U16 setting}</WriteAnalog>	
Example	<WriteAnalog> 1; 344 </WriteAnalog>	
Arguments	portNumber	Analog output port identifier
	Value range	0 = Laser Power port (LASER_ANALOG0) 1 = Auxiliary Analog output port (LASER_ANALOG1)
	setting	New port value
	Value range	0 - 4095

WriteDigital	
Description	Sets the digital output port to a new value.
Syntax	<WriteDigital>{U16 portNumber; U16 setting}</WriteDigital>
Example	<WriteDigital> 3; 1 </WriteDigital>

WriteDigital																						
Arguments	portNumber	<div>Port identifier</div> <table><tr><td>Port</td><td>Association</td></tr><tr><td>0</td><td>AUX_JOBACTIVE</td></tr><tr><td>1-4</td><td>AUX_GPO1-4</td></tr><tr><td>5</td><td>AUX_LASING</td></tr><tr><td>6</td><td>AUX_READY</td></tr><tr><td>7</td><td>AUX_BUSY</td></tr><tr><td>16-31</td><td>Extended DOUT bits 0-15</td></tr><tr><td>100</td><td>System status ports as a group</td></tr><tr><td>101</td><td>Extended I/O ports as a group</td></tr><tr><td>102</td><td>8-bit digital power port</td></tr></table>	Port	Association	0	AUX_JOBACTIVE	1-4	AUX_GPO1-4	5	AUX_LASING	6	AUX_READY	7	AUX_BUSY	16-31	Extended DOUT bits 0-15	100	System status ports as a group	101	Extended I/O ports as a group	102	8-bit digital power port
	Port	Association																				
	0	AUX_JOBACTIVE																				
	1-4	AUX_GPO1-4																				
	5	AUX_LASING																				
6	AUX_READY																					
7	AUX_BUSY																					
16-31	Extended DOUT bits 0-15																					
100	System status ports as a group																					
101	Extended I/O ports as a group																					
102	8-bit digital power port																					
	Value range	See port numbers (above).																				
	setting	<div>The new value for the port (as an unsigned 16-bit integer)</div> <div>Note: The actual signal polarity is determined by how the optical isolators are connected.</div> <div>Single bit mode (ports 0-31):</div> <div>0 (unasserted) and 1 (asserted)</div> <div>Group mode (ports 100-102)</div> <div>0 (unasserted) and 1 (asserted) in bit positions defined as follows:</div> <table><tr><td>Port</td><td>Bits</td><td>Signal</td></tr><tr><td rowspan="2">100</td><td>0-3</td><td>AUX_GPO1-4</td></tr><tr><td>4-7</td><td>AUX_LASING, AUX_BUSY, AUX_READY, AUX_JOBACTIVE</td></tr><tr><td>101</td><td>0-15</td><td>Extended DOUT bits 0-15</td></tr><tr><td>102</td><td>0-7</td><td>Laser digital bits 0-7</td></tr></table>	Port	Bits	Signal	100	0-3	AUX_GPO1-4	4-7	AUX_LASING, AUX_BUSY, AUX_READY, AUX_JOBACTIVE	101	0-15	Extended DOUT bits 0-15	102	0-7	Laser digital bits 0-7						
Port	Bits	Signal																				
100	0-3	AUX_GPO1-4																				
	4-7	AUX_LASING, AUX_BUSY, AUX_READY, AUX_JOBACTIVE																				
101	0-15	Extended DOUT bits 0-15																				
102	0-7	Laser digital bits 0-7																				
	Value range	0 - 65535																				

6.5.7 UTILITY COMMANDS

ApplicationEvent		
Description	<p>Defines an application event.</p> <p>Note: Application events are used to notify the application that a certain point in the execution of the job has been reached. Events are handled by the application using the OnMessageEvent method.</p> <p>Application events should be used sparingly as system performance could be affected if they are generated at a high rate.</p>	
Syntax	<ApplicationEvent>{U16 param1; U32 param2}</ApplicationEvent>	
Example	<ApplicationEvent> 100; 345 </ApplicationEvent>	
Arguments	param1	First application-specific parameter. Value is returned in OnMessageEvent puiPayloadHigh [31..16].
	Value range	0 - 65536
	param2	Second application-specific parameter. Value is returned in OnMessageEvent puiPayloadLow [31..0]
	Value range	0 - 2 ³² -1

BeginJob		
Description	<p>Generates a BeginJob application event when executed by the marking engine. The JobDataCnt parameter in the StatInfoData broadcast packet is re-initialized to zero. BeginJob automatically sets the system BUSY signal.</p>	
Syntax	<BeginJob></BeginJob>	
Example	<BeginJob></BeginJob>	
Arguments	N/A	N/A
	Value range	N/A

EndJob		
Description	Generates an EndJob application event when executed by the marking engine. The system BUSY and MARKINPROG signals are automatically cleared.	
Syntax	<EndJob></EndJob>	
Example	<EndJob></EndJob>	
Arguments	N/A	N/A
	Value range	N/A

GalvoCmdMarker		
Description	Inserts a marker into the Lightning II command stream via the GSBUS. The marker can be used to synchronize Lightning II probe data collection	
Syntax	<GalvoCmdMarker></GalvoCmdMarker>	
Example	<GalvoCmdMarker></GalvoCmdMarker>	
Arguments	N/A	N/A
	Value range	N/A

JobDataCntr	
Description	<p>Sets the job data counter to the specified value.</p> <p>Note: The job data counter is incremented as each 32-bit data element of the job stream is processed by the marking engine. This is useful for tracking how much data the marking engine has processed at any given time. The current value of the counter is reported in the System Status broadcast data as element name <u>JobDataCntr</u>.</p>
Syntax	<JobDataCntr>{U32 value}</JobDataCntr>

JobDataCntr		
Example	<JobDataCntr>0</JobDataCntr>	
Arguments	value	Counter value
	Value range	0 (only accepts zero for now)

JobMarker		
Description	<p><i>(Reserved for future use).</i> Generates an Application Event of the type <u>MarkProgress</u> and/ or <u>CycleProgress</u> as the job progresses. The current JobMarker data value is also available in the broadcast status data <u>JobMarker</u> element.</p> <p>Note: The Application Events generated by this command are typically used to track job execution progress.</p>	
Syntax	<JobMarker>{U16 value}</JobMarker>	
Example	<JobMarker>35</JobMarker>	
Arguments	value	<p>An encoded, application-defined marker value. Bits[7..0] encode the <u>MarkProgress</u>, and Bits[14..8] encode the <u>CycleProgress</u>. The marker value is sampled every 200msec by the SMC. If the value changes within this interval, an <u>ApplicationEvent</u> will be generated.</p> <p>If bit[15] of the most recent JobMarker value is set, a <u>MarkProgress</u> message will be generated. Otherwise, a <u>CycleProgress</u> message will be generated.</p> <p>The corresponding JobMarker field data—right-shifted in the case of a <u>CycleProgress</u> message—will be returned in the <u>puiPayloadLow</u> value of the OnMessageEvent.</p>
	Value range	0 - 65535

JobTimer		
Description	Controls the state of the hardware job timer. Used to time actual execution time of a job. The last saved timer value is returned as the JobTimer value in the XML data packet returned when using the <u>GetRegisters</u> priority message.	
Syntax	<JobTimer>{U16 action}</JobTimer>	
Example	<JobTimer>1</JobTimer>	
Arguments	action	Controls the state of the timer logic.
	Value range	0 = Clear the timer but do not count 1 = Start or continue the timer 2 = Stop or pause the timer 3 = Save the timer (a snap-shot of the timer is taken for readback)

LongDelay		
Description	Delays all operations for the specified duration.	
Syntax	<LongDelay>{U32 duration}</LongDelay>	
Example	<LongDelay>10000</LongDelay>	
Arguments	duration	Length of time to sleep (in μsec s)
	Value range	$0 - (2^{32}-1)/100 \rightarrow (0 - 42,949,672)$

Set		
Description	Sets the named parameter to the specified value.	
Syntax	<Set id='{STR parameter}'>{value(s)}</set>	
Example	<Set id='MarkSpeed'>10; 2</set>	
Arguments	parameter	The name of the parameter to be set
	Value range	Any valid parameter name

Set		
	value(s)	The number of arguments is specific to the named parameter.
	Value range	The values of the argument(s) is specific to the named parameter.

6.5.8 COORDINATE SYSTEM TRANSFORM PARAMETERS

ActiveCorrectionTable		
Description	Sets the active current correction table.	
Syntax	<set id='ActiveCorrectionTable'>{U16 table}</set>	
Example	<set id='ActiveCorrectionTable'>1</set>	
Arguments	table	Identifies the correction table to be used
	Value range	<p>1 – 4, 257- 260</p> <p>Note: Only Tables 1 and 2 should be selected during job execution. Tables 3 and 4 are used only for loading alternate data for the XY2-100 interface and for GSBUS channels 3 - 5. See <i>Figure 26 - Multiple Correction Table Usage in the SMC</i> on page 293 for more details.</p> <p>Note: The secondary values 257 – 260 are used to indicate that correction table data being sent to the controller using a subsequent <u>sendStreamData</u> method is either 20-bit (for EC1000 platforms with 20-bit firmware) or 24-bit for the SMC. This represents setting the 0x100 bit in the <i>table</i> value. Setting this bit is only necessary when sending a table using sendStreamData(), not when switching between tables at run-time.</p>

FieldOffset		
Description	Sets the offsets to be applied to vectors at run-time. These offsets are integrated into the position commands during micro-vector operations. These values override the offsets specified in the <u>UserConfig</u> and <u>LensConfig</u> files.	
Syntax	<set id='FieldOffset'>{FLT xOffset; FLT yOffset; FLT zOffset}</set>	
Example	<set id='FieldOffset'> 5000; -1000; 100 </set>	
Arguments	xOffset	Offset to be applied to the X vector at run-time.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the X virtual field size of the system in floating point notation.
	yOffset	Offset to be applied to the Y vector at run-time.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Y virtual field size of the system in floating point notation.
	zOffset	Offset to be applied to the Z vector at run-time. Note: The Z offset is optional. If the Z offset is not specified, it is set to zero.
	Value range	$2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Z virtual field size of the system in floating point notation.

FieldOrientation	
Description	Sets the orientation of the marking field relative to the vector coordinate system. This transformation is applied at run-time. This value overrides the <i>Rotation</i> setting contained in the <u>UserConfig</u> file.
Syntax	<set id='FieldOrientation'>{U16 rotation}</set>
Example	<set id='FieldOrientation'> 90 </set>

FieldOrientation		
Arguments	rotation	Specifies the counter-clockwise rotation of the marking field in degrees.
	Value range	0, 90, 180, 270

Offset		
Description	Sets the offsets to be applied to the vector set before being passed to the SMC. These offsets are integrated in to the job data as it is being compiled. A saved job will have these offsets built into it.	
Syntax	<set id='Offset'>{FLT xOffset; FLT yOffset; FLT zOffset}</set>	
Example	<set id='Offset'> 1000; 2000; 100 </set>	
Arguments	xOffset	Offset in bits to be applied to the X vector at run-time.
	Value range	$2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the X virtual field size of the system in floating point notation.
	yOffset	Offset in bits to be applied to the Y vector at run-time.
	Value range	$2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Y virtual field size of the system in floating point notation.
	zOffset	Offset in bits to be applied to the Z vector at run-time. Note: The Z offset is optional. If the Z offset is not specified, it is set to zero.
	Value range	$2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Z virtual field size of the system in floating point notation.

HeadOffset		
Description	Set offsets on a per-scanhead basis.	
Syntax	<set id='HeadOffset'>{U16 headID, FLT xOffset; FLT yOffset [; FLT zOffset]}</set>	
Example	<set id='HeadOffset'> 0; 1000; 2000; 100 </set>	
	headID	Scanhead number to apply the offset values to
	Value range	0 - 1
	xOffset	Offset in <i>units</i> to be applied to the specified scanhead X axis at run-time.
	Value range	$2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the X virtual field size of the system in floating point notation.
	yOffset	Offset in <i>units</i> to be applied to the specified scanhead Y axis at run-time.
	Value range	$2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Y virtual field size of the system in floating point notation.
	zOffset	<i>Reserved for future use.</i> Offset in <i>units</i> to be applied to the specified scanhead Z axis at run-time. Note: The Z offset is optional. If the Z offset is not specified, it is set to zero.
	Value range	$2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Z virtual field size of the system in floating point notation.

Transform	
Description	Sets the values of the coordinate transform matrix to be applied to the vector set before being passed to the SMC.

Transform		
Syntax	<set id='Transform'>{FLT M00; FLT M01; FLT M10; FLT M11}</set>	
Example	<set id='Transform'> 1.0; 0.0; 0.0; 1.0 </set>	
Arguments	M00 – M11	Represents the four transformation matrix elements $\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} M00 & M01 & 0 \\ M10 & M11 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$
	Value range	Any floating point value

TransformEnable		
Description	Enables or disables run-time coordinate transformations using the transform selected by the ID argument. The transform information is specified using the Priority data message SetRTJobTransform2D .	
Syntax	<Set id='TransformEnable'>{U16 transformID}</Set>	
Example	<Set id='TransformEnable'>1</Set>	
Arguments	transformID	Enables a specific run-time coordinate transformation or disables all run-time coordinate transformations.
	Value range	0 = All run-time coordinate transformations are disabled 1 = Use transform ID 1 2 = Use transform ID 2

RTCCompatibility	
Description	Enables or disables coordinate system compatibility with the Scanlab RTC controller family. Enabling RTC compatibility causes the X and Y axes to be swapped after the correction table calculations are done, effectively causing the field to rotate 90 degrees counter-clockwise. If the Scanlab mode is used, then Scanlab .ctb correction files can be used directly via the session method sendCorrectionData .
Syntax	<Set id='RTCCompatibility'>{BOOL state}</Set>

RTCCompatibility		
Example	<Set id='RTCCompatibility'> true </Set>	
Arguments	state	Specifies the “enabled” or “disabled” state
	Value range	<p>true = Coordinate system compatibility with the Scanlab RTC controller family is enabled.</p> <p>false =Coordinate system compatibility with the Scanlab RTC controller family is disabled.</p>

HeadTransform		
Description	Sets the values of the coordinate transform matrix to be applied to each head’s command data prior to the correction table. This transform operates on micro-vector data. Both heads can have separate transforms.	
Syntax	<set id='HeadTransform'>{U16 headID, FLT M00; FLT M01; FLT M10; FLT M11}</set>	
Example	<set id='Transform'> 1.0; 0.0; 0.0; 1.0 </set>	
Argument s	headID	Scanhead number to apply the offset values to
	Value range	0 - 1
	M00 – M11	<p>Represents the four transformation matrix elements</p> $\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} M00 & M01 & 0 \\ M10 & M11 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$
	Value range	Any floating point value within this range: Value > -2.0 and Value < 2.0

6.5.9 HARDWARE INTERFACE CONFIGURATION PARAMETERS

These configuration parameters are set in the configuration files stored on the SMC and automatically applied at power-up. They are available here to permit overriding those settings.

SMCInsGenMode		
Description	Sets the method for computing galvo path trajectories.	
Syntax	<set id='SMCInsGenMode'>{U16 mode}</set>	
Example	<set id='SMCInsGenMode'>0</set>	
Arguments	mode	The trajectory planning method used at run-time to create marks and jumps
	Value range	0 – Traditional Mode velocity step trajectories 1 – ScanPack mode acceleration/jerk limited trajectories. Limited to use with SMAPI only.

AxisDACConfig (Obsolete)	
Description	<p>Sets the analog command output configuration for the laser galvo servo controllers using a bitmask.</p> <p>Note: This is normally set in the Controller Configuration file, but can be overridden with this command.</p>
Syntax	<pre><set id='AxisDACConfig'>{<i>HEX bitmask</i></set></pre>

AxisDACConfig (Obsolete)		
Example	<code><set id='AxisDACConfig'>0x6</set></code>	
Arguments	bitmask	Bitmask which defines analog output configuration
	Value range	<p>The mask is defined as follows:</p> <p>Bits 1..0 encode the range of the X & Y DACs.</p> <p>Bits 3..2 encode the range of the Z DAC.</p> <p>The single-ended voltage range encoding is as follows:</p> <p>00 = $\pm 2.5V$, 01 = $\pm 5V$, 10 = $\pm 10V$</p>

LaserModeConfig		
Description	Sets the laser configuration bitmask. A default value can be set in the Laser Configuration file as the parameter <u>LaserModeConfig</u> .	
Syntax	<set id='LaserModeConfig'> <i>{HEX bitmask}</i> </set>	
Example	<set id='LaserModeConfig'> 0x1FF </set>	
Arguments	bitmask	Bitmask which defines the laser configuration
	Value range	The bit definitions for the bitmask are shown below.

LaserModeConfig				
		Name	Hex Bit Value	Definition
		LASER_GATE polarity	0x0001	0=active high, 1=active low
		LASER_POINTER polarity	0x0002	0=active high, 1=active low
		Laser Sync Mode Bit 0	0x0004	See notes below
		LASER_MOD1 polarity	0x0008	0=active high, 1=active low
		LASER_MOD2 polarity	0x0010	0=active high, 1=active low
		LASER_MOD3 polarity	0x0020	0=active high, 1=active low
		LASER_ENABLE polarity	0x0040	0=active high, 1=active low
		LASER_DOUT polarity	0x0080	0=active high, 1=active low
		Laser activate	0x0100	1=activate (enable) laser output signals
		Laser Power Port mode	0x0200	Set the mode of the digital laser power port 0=8-bit mode, 1=7-bit mode (LSB used as strobe)
		LASER_POINTER configuration	0x0400 & 0x0800	Sets the mode of operation of LASER_POINTER

LaserModeConfig				
				0 - LASER_POINTER == NOT LASER_GATE 1 - LASER_POINTER == LASER_GATE AND NOT LasersEnabled 2 - LASER_POINTER == NOT LasersEnabled 3 - LASER_POINTER == Asserted all of the time
		Laser Power Port	0x1000	0=8-bit digital power port, 1=analog output A1
		LASER_GATE configuration	0x2000	0=Gating signal, 1=Modulation signal if 8-bit digital power port bit 7 is also set.
		LASER_GATE inhibit	0x4000	0=normal operation, 1=LASER_GATE is suppressed when the laser is turned on but the modulation signal is still emitted. Use in synchronous laser operation during <u>JumpAndFireList</u> commands.
		Laser Sync Mode Bit 1	0x8000	See notes below.
		Notes on Laser Sync Mode: Laser Sync Mode bits [1 – 0] encode the laser synchronization mode of the SMC according to the following table:		

LaserModeConfig		
		0 = Asynchronous modulation. The laser modulation is discontinuous, switching between the background modulation and the lasing modulation coincident with the LASER_GATE signal
		1 = Synchronous to the modulation signal on LASER_MOD3. LASER_MOD3 takes its modulation settings from the background settings for LASER_MOD1. The background signal for LASER_MOD1 and LASER_MOD2 is set for no modulation. In this mode, the LASER_GATE and subsequent LASER_MOD1 and LASER_MOD2 timing is synchronized to the rising edge of pulses on LASER_MOD3
		2 = Synchronous to the free-running modulation of LASER_MOD2. In this mode the LASER_GATE signal is synchronized to the falling edge of LASER_MOD2. Both LASER_MOD1 and LASER_MOD2 are free-running according to the LaserPulse settings defined for them.
		3 = Synchronous to the external signal source received on LASER_STAT6. In this mode, the LASER_GATE and subsequent LASER_MOD1 and LASER_MOD2 timing is synchronized to the rising edge of pulses received on LASER_STAT6.

ServoConfig (Obsolete)		
Description	Sets the configuration of the X, Y and Z servo control interface.	
Syntax	<set id='ServoConfig'> <i>{HEX bitmask}</i> </set>	
Example	<set id='ServoConfig'> 0x4 </set>	
Arguments	bitmask	Bitmask which defines the configuration of the laser galvo servo interface

ServoConfig (Obsolete)				
	Value range	The bit definitions for the bitmask are shown below.		
		Name	Hex Bit Value	Definition
		X_SERVO_EN and Y_SERVO_EN polarity	0x0001	0=active high, 1=active low
		Z_SERVO_EN polarity	0x0002	0=active high, 1=active low
		Enable X, Y Servos	0x0004	1=enable servos, 0=disable
		Enable Z Servo	0x0008	1=enable servos, 0=disable
		X_SERVO_RDY and Y_SERVO_RDY polarity	0x0010	0=active high, 1=active low
		Z_SERVO_RDY polarity	0x0020	0=active high, 1=active low
		X, Y Not-ready exception enable	0x0040	1=enable exception event generation is X or Y servo becomes not ready
		Z Not-ready exception enable	0x0080	1=enable exception event generation if Z servo becomes not ready

6.5.10 BIT-MAP RASTER SUPPORT

Bit-map raster rendering can be performed in four different modes depending on the level of quality and throughput required. Two “fire-on-the-fly” modes and two “step-and-shoot” modes are supported. These modes are illustrated in the following figures that show the relative galvo motion and laser modulation control.

Mode 0: Variable Pulse Width "Fire-on-the-fly"

Mode 0 raster patterning permits gray-scale imaging when the laser supports variable laser power as a function of how long the laser modulation signal remains on. This is typical of how CO2 lasers operate. In this illustration, the “high” pulse-width is proportional to an 8-bit gray-scale pixel value. Since the laser fires at a constant rate and the start of the galvo position commands and the start of the lasing process is tightly controlled, the start of each pixel position is accurately placed on the substrate.

This mode is also useful in a thresholded or error-diffusion dithering gray-scale approximation approach using Q-Switched lasers. In this case, a low-thresholded or “0” pixel value can cause the pulse-width to be set to 0 thus skipping the firing of the laser at that pixel location. Likewise, a high-thresholded or “1” pixel value can cause the laser to fire at that location.

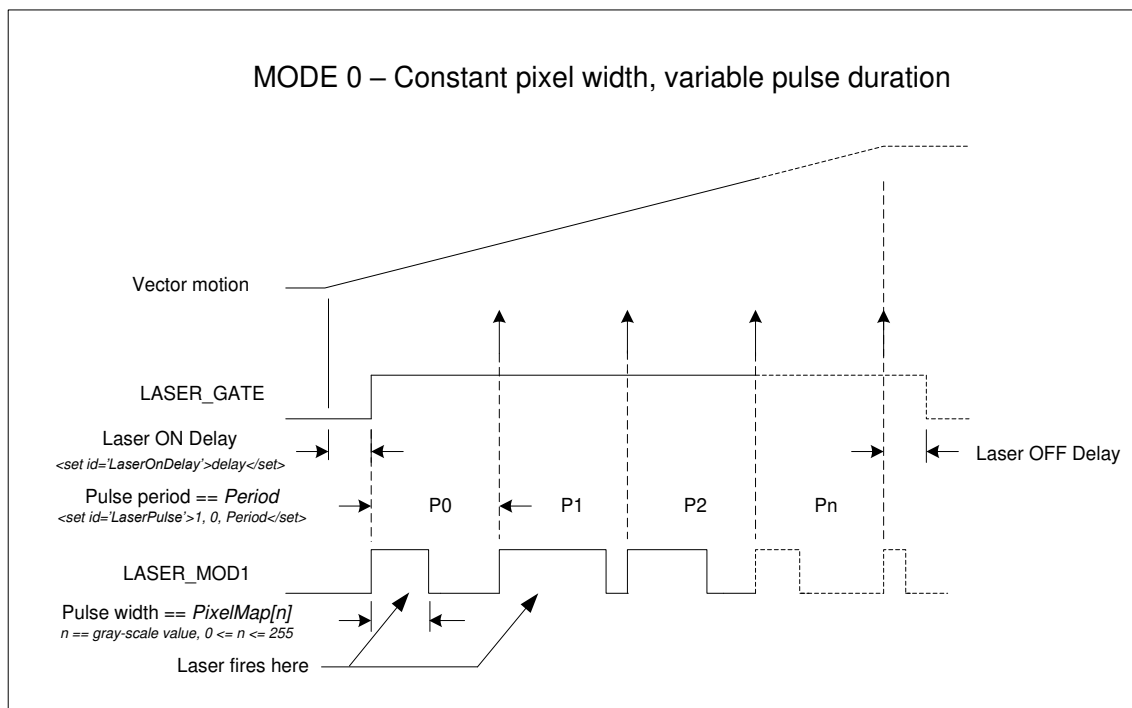


Figure 4 - “FIRE-ON-THE-FLY”, MODE 0

Mode 1: Variable Power “Fire-on-the-fly”

Mode 1 raster patterning permits gray-scale imaging when the laser supports variable laser pulse power as a function of variable analog or digital laser power control. In this illustration, the laser power control is set proportional to an 8-bit gray-scale value at the beginning of a pixel period, and the laser fires at the end of the period on each rising edge of the laser modulation signal. The pulse width of the laser modulation signal is programmable and stays the same for each pixel in the pixel line. Since the laser fires at a constant rate and the start of the galvo position commands and the start of the lasing process are tightly controlled, the pixels positions are accurately placed on the substrate.

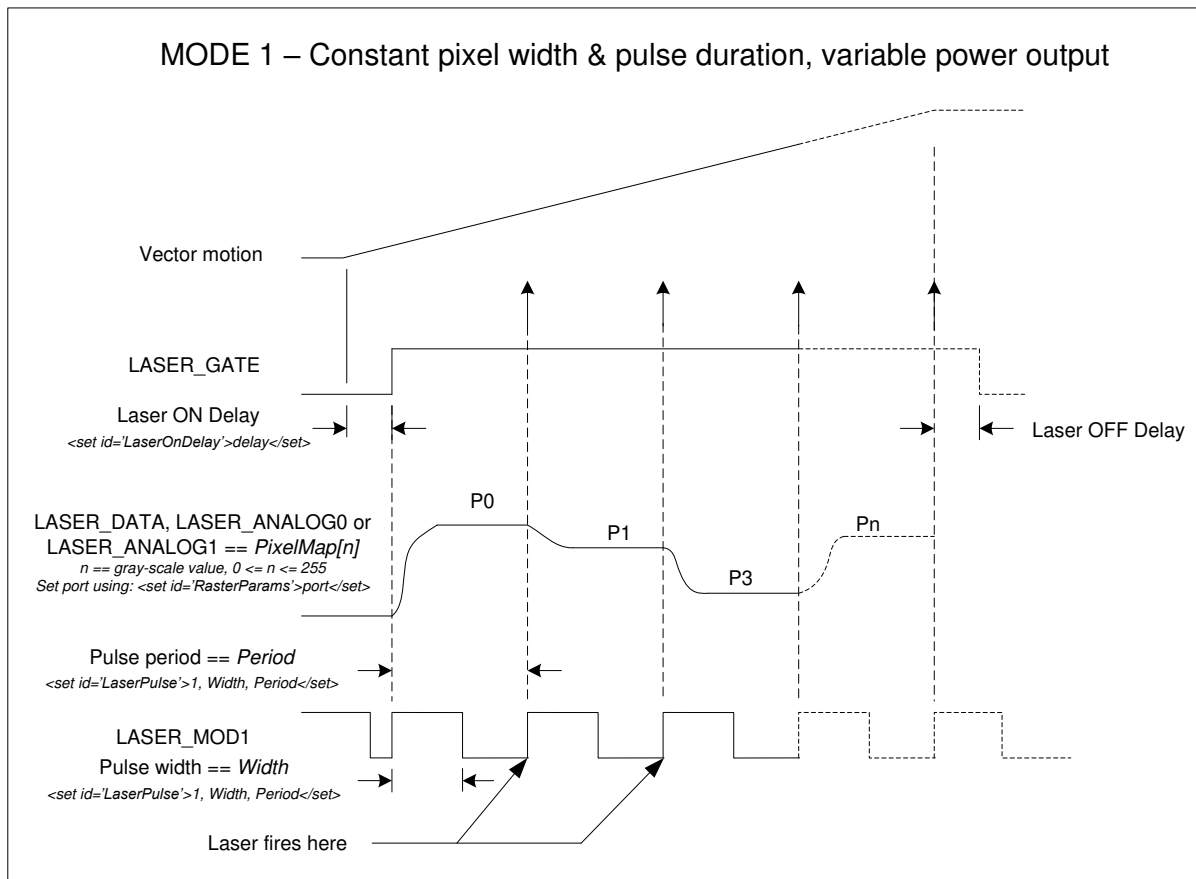


Figure 5 - “FIRE-ON-THE-FLY”, MODE 1

Standard Jump-and-fire Raster Mode

Jump-and-fire raster patterning permits very accurate gray-scale imaging with most CO2 lasers, and gray-scale approximations using pulsed YAG lasers. For CO2 lasers, gray scale is achieved by controlling the pulse width of the modulation signal when the laser fires at a pixel location. The galvos are instructed to jump to each pixel location, a LaserOnDelay time is incurred to let the galvos settle, and then the laser fires for the specified LaserOnTime. One or several pulses may be emitted at each pixel, depending on the pulse period specified with the <set id='LaserPulse'> command.

If the LaserPulse period is set to be the same as the LaserOnTime, then a single pulse will be emitted at each pixel. With fast CO2 lasers, this provides variable laser power proportional to the pulse width. For pulsed YAG lasers used to expose single dots per pixel using error diffusion methods, the pulses can be suppressed with a pulse-width value of zero, or fired with an appropriate non-zero value.

Since the galvos jump to each pixel location and stop there before firing, very precise pixel placement is achieved regardless of the scanning direction. Precision can be increased by lengthening the LaserOnDelay parameter but at the cost of some speed.

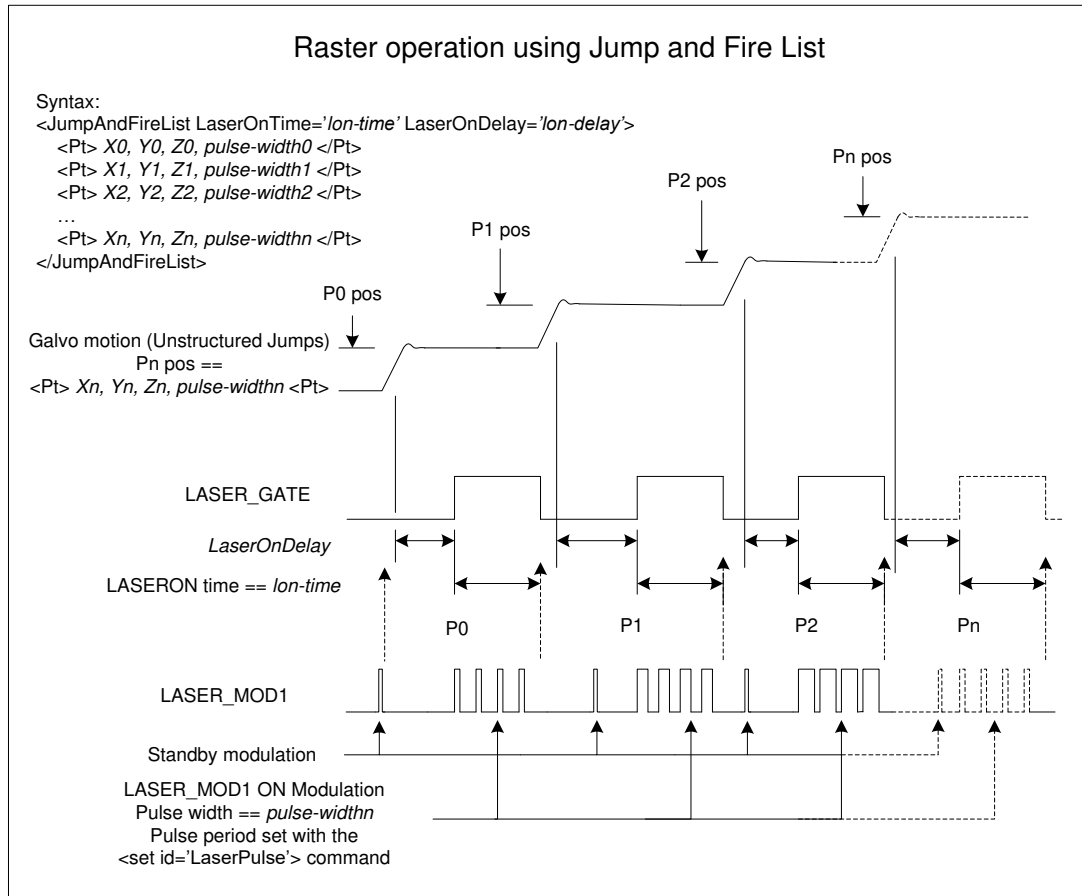


Figure 6 - STANDARD “JUMP-AND-FIRE” MODE

Synchronous Fiber Laser Jump-and-fire Raster Mode

Some new fiber lasers require continuous modulation to which the laser firing circuitry phase-locks. Firing the laser requires the assertion of the LASER_GATE signal in precise timing relationship to a constantly emitted pulse train. Other similar lasers require that the modulation sequence be provided by the laser and that pulses "picked" when the laser is intended to fire. Both modes of operation are supported by the SMC.

Setting the Laser Sync Mode [1 – 0] bits to the value 2 using the job command `<set id=LaserModeConfig>` causes the SMC hardware to change its laser control behavior to constantly emit laser pulses on the LASER_MOD1 signal according to the `<set id='LaserPulse'>` parameters. All subsequent laser and galvo operations are then synchronized to the pulse train. If however the Laser Sync Mode [1 – 0] bits are set to the value 3, then the pulse stream is taken from the SMC

LASER_STAT6 digital input. This permits synchronization to lasers that create their own pulse generating signal.

In the *JumpAndFireList* command, the *OutputMode* attribute selects how to use the pixel value. By default, the laser pulse-width is changeable on a pixel-by-pixel basis with a special case for a pixel value of zero. For non-zero pixel values, the pulse width is set to the pixel value (in laser-ticks) and the LASER_GATE signal is asserted for the time (in laser-ticks) specified by the *LaserOnTime* attribute of the *JumpAndFireList* command. If the pixel value is zero, then the LASER_GATE signal is suppressed during the *LaserOn* interval. Even though the LASER_GATE signal is suppressed, the *LaserOnDelay* and *LaserOnTime* intervals are present resulting in a consistent pixel time. The overall result is that the laser can retain phase-lock and be selectively fired on a pixel-by-pixel basis.

The *OutputMode* attribute can specify any of the following alternate targets for the pixel data:

- 0 = pulse-width (default)
- 1 = LASER_ANALOG0 (analog power port)
- 2 = LASER_ANALOG1
- 3 = LASER_DATA (digital power port)

In the case of the analog output ports, 12 bits of resolution are supported, whereas only 8 bits are supported for the digital power port. In these non-default output modes, the pixel data is applied to the port after the jump but before the *LaserOnDelay* attribute value is applied. After the *LaserOnDelay*, the laser will fire per the settings specified by the `<set id='LaserPulse'>` command, but synchronous with the next pulse in the pulse-train. If *LaserModSyncSrc* is *Int*, then the LASER_GATE signal will be synchronous with the leading edge of the next pulse. If *LaserModSyncSrc* is *Ext*, then the LASER_GATE signal will be synchronous with the falling edge of the LASER_STAT6_ISO signal.

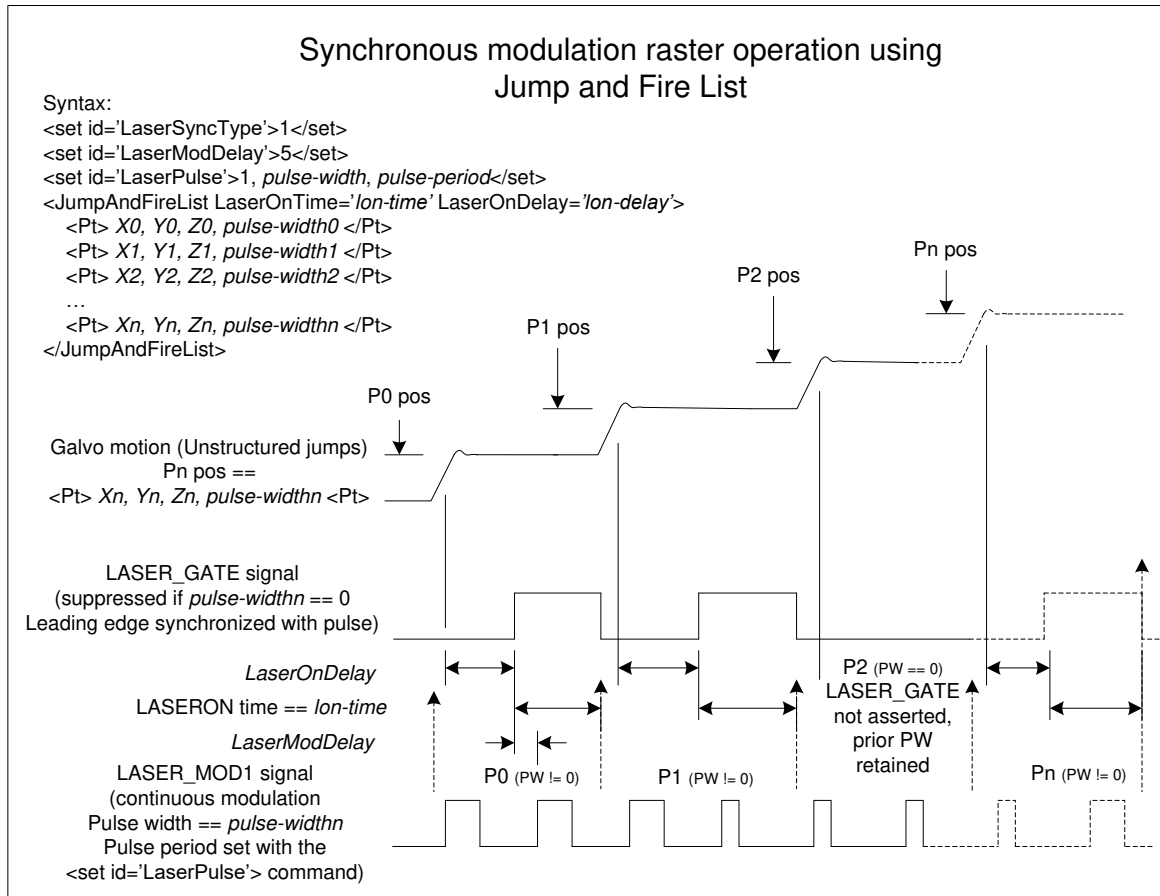


Figure 7 - SYNCHRONOUS “JUMP-AND-FIRE” MODE

6.5.11 BIT-MAP RASTER COMMANDS

Raster operations are defined through the use of the commands defined in the following section (“**Error! Reference source not found.**”). These commands can be freely placed anywhere in a job.

The API supports a pixel mapping table that permits non-linear mapping of 8-bit pixel values to the appropriate laser control values required by the selected mode. This permits a linear range of gray-scale pixel values to scale into a range that is appropriate for the behavior of the laser and materials being used.

Bit-Map Raster Parameters and Commands

RasterMode		
Description	Selects the mode of raster operation.	
Syntax	<set id='RasterMode'>{U16 mode}</set>	
Example	<set id='RasterMode'>1</set>	
Arguments	mode	raster mode
	Value range	0 = Variable Pulse Width "Fire-on-the-fly" 1 = Variable Power "Fire-on-the-fly"

PixelMap								
Description	Sets the values of the pixel mapping table.							
Syntax	<set id='PixelMap'>{U16 PM0; U16 PM1; ... U16 PM255}</set>							
Example	<set id='PixelMap'> 0; 1; 2; ... 255 </set>							
Arguments	PMn	256 entries are used to form a table that is indexed by the actual gray-scale pixel value specified in a <u>RasterLine</u> command. The table value indexed by the gray-scale pixel value represents the variable part of laser control system per the selected raster mode. <table><tr><td>Mode</td><td>Table value interpretation</td></tr><tr><td>0</td><td>Laser ON pulse width</td></tr><tr><td>1</td><td>Laser power control</td></tr></table>	Mode	Table value interpretation	0	Laser ON pulse width	1	Laser power control
	Mode	Table value interpretation						
	0	Laser ON pulse width						
1	Laser power control							
	Value range	0 - 255						

RasterParams	
Description	Sets mode-specific parameters.
Syntax	<set id='RasterParams'>{U16 param}</set>
Example	<set id='RasterParams'>1</set>

RasterParams			
Arguments	param	Mode 1	Parameter Selection Pixel output port selection 0 = LASER_ANALOG0 1 = LASER_ANALOG1 2 = LASER_DATA (8-bit digital port)
	Value range	0 - 2	

RasterLine		
Description	Specifies the data and trajectory of a raster line.	
Syntax	<RasterLine X='{FLT xDest}' Y='{FLT yDest}' Z='{FLT zDest}'>{U8 P0; U8 P1; ... U8 Pm}</RasterLine>	
Example	<RasterLine X='10000' Y='0' Z='0'>25;15;44;0;0;33;34, ...</RasterLine>	
Arguments	xDest yDest zDest	X, Y, Z coordinate of the end of the raster line. Values are floating point and are converted into system "bits" units per the <u>Units</u> command.
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, these values can take on a range which represents the X, Y, Z virtual field size of the system in floating point notation.
	P0 - Pn	A list of 8-bit pixel values to be exposed along the line. The values are used to index the PixelMap table to fetch an actual laser power level control that will be set at each pixel location. A maximum of 65535 pixels per line can be specified.
	Value range	0 - 255

6.5.12 POLYGON BIT-MAP RASTER COMMANDS

Polygon raster operations are defined through the use of the commands defined in the following section ("Polygon Bit-Map Raster Parameters and Commands"). These commands can be freely placed anywhere in a job after calculating parameters based on a polygon device configuration file.

Because of the high rates of speed in a polygon system, marking is performed by firing (or not) a single laser pulse at each pixel location. Laser power control is static for the entire pixel line. The implementation is similar to Raster Mode 0 where pixels are fired on-the-fly, but without pixel-level power control.

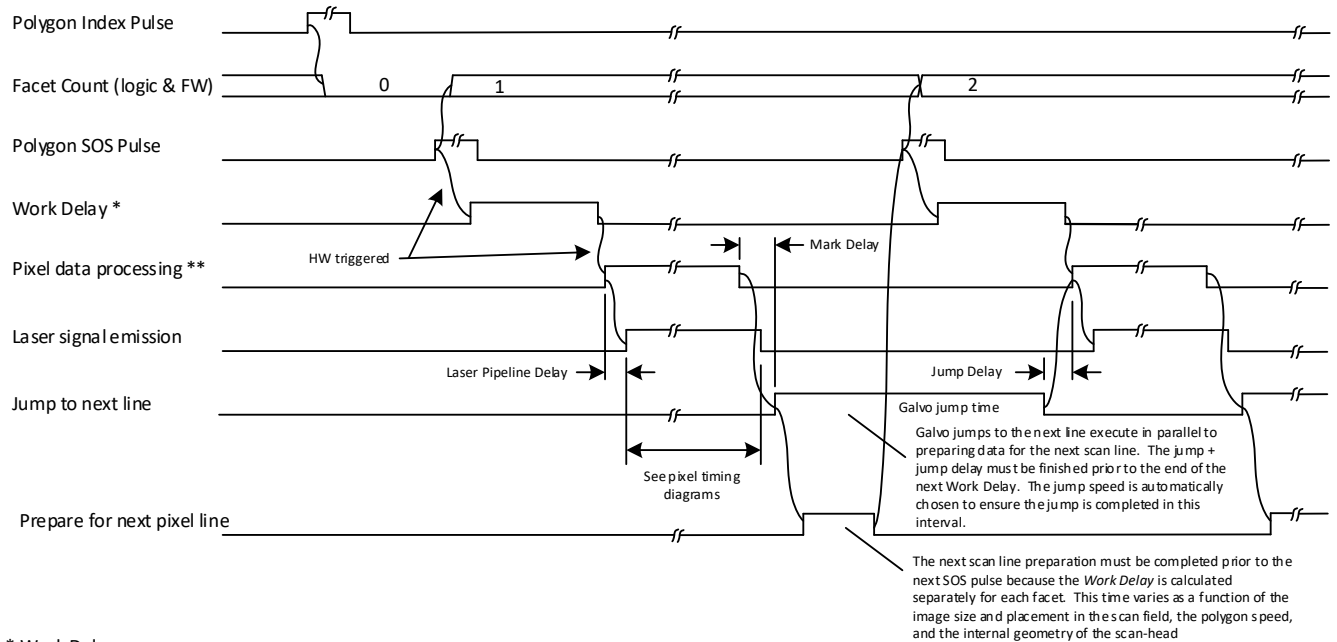
To create the fire (or not) pixel data, the 8-bit pixel data specified in the raster line represents a packed value of 8 pixels, one bit per pixel. The least significant bit is emitted first, working right to left to the most significant bit. Sequences of four RasterLine pixel values are processed together to form a 32-pixel entity. A complete raster line should contain a multiple of four pixel entries. The API will pad with non-firing pixel data at the end of the line if insufficient pixel entries are provided.

Gray-scale approximation can be accomplished by using error diffusion dithering techniques at the application level. This capability is not directly supported by this API.

Polygon Bit-Map Raster Parameters and Commands

Polygon operations require precise synchronization of the polygon position, corrective galvo operation, and laser modulation. The following diagram shows the relevant timing relationships.

Session API

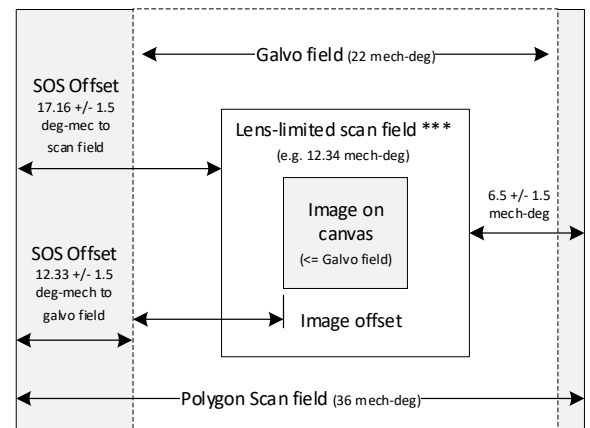


- * Work Delay = SOS delay + Image offset delay + Facet delay + Inset delay
- SOS delay is calculated from SOS angle offset to the edge of the Galvo field, and the polygon scan speed (lines-per-sec) value
 - Image offset delay is calculated from job layout and Mark Speed ***
 - Facet delay is calculated from the line-start shift calibration measurements in micron units
 - Inset delay is calculated as a % of pixel period

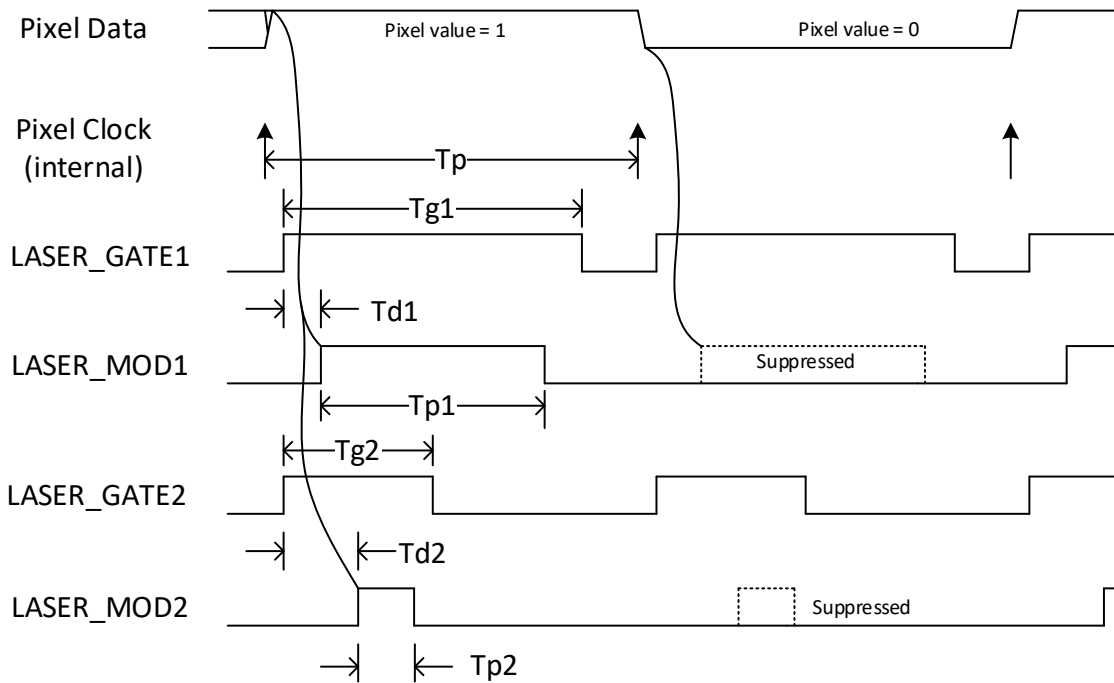
** In the pixel data processing interval, the X galvo is concurrently applying correction table value in the direction of the scan

*** The scan field size can be smaller than the galvo field size due to the choice of scan lens. The galvo field size is defined in the correction table file using the galvo ½ mechanical angle property. The field size in mm is defined by the system X-Y calibration factors (bits/mm) where the bits range is 2^{24} . The calibration factors are initially derived from the lens focal length and galvo range, but are adjusted as needed during calibration.

**** Mark Speed (mm/sec) is calculated from the system X-Y calibration factors, the galvo field size and the polygon scan speed (lines-per-sec) value



There are two laser modulation schemes supported when polygon operation is enabled (see **Error! Reference source not found.** section). These are shown in the following figures.



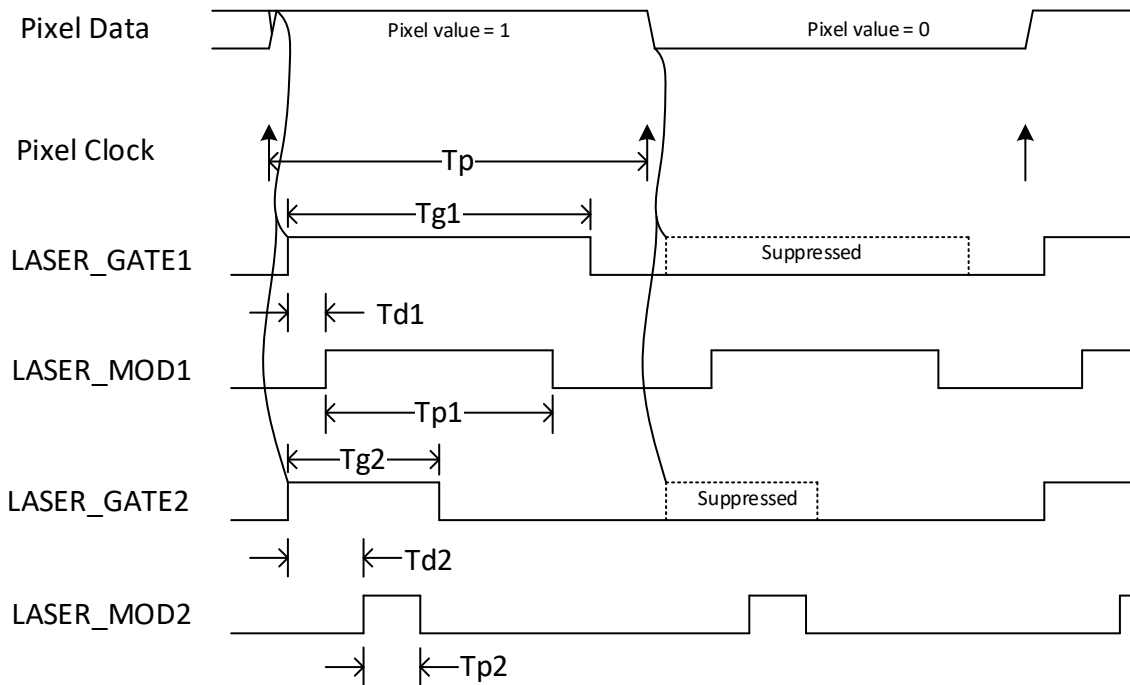
Modulation Mode 0: Gated triggering

T_p = Pixel Period. Dependent on polygon speed and output DPI. Internally calculated.

T_{g1} & T_{g2} = Pixel Gate signal. Starts after Pixel Clock. Recommend length be specified as % of pixel period. May be made continuous if 100% of Pixel Period.

T_{p1} & T_{p2} = Laser Trigger Pulse. Suppressed if pixel value == 0. Recommend length be specified as % of Pixel Gate length

T_{d1} & T_{d2} = Laser Trigger Pulse Delay. Units of Nano Seconds rounded up/down to nearest 10 Nano Seconds.



Modulation Mode 1 – Continuous triggering with selective gating

T_p = Pixel Period. Dependent on polygon speed and output DPI. Internally calculated.

T_{g1} & T_{g2} = Pixel Gate signal. Starts after Pixel Clock. Suppressed if pixel value == 0. Recommend length be specified as % of pixel period

T_{p1} & T_{p2} = Laser Trigger Pulse. Always present. Recommend length be specified as % of Pixel Gate length

T_{d1} & T_{d2} = Laser Trigger Pulse Delay. Units of Nano Seconds rounded up/down to nearest 10 Nano Seconds.

The following commands are used to configure polygon operation.

PolygonMode		
Description	Selects the mode of polygon raster operation.	
Syntax	<set id='PolygonMode'>{U16 mode}</set>	
Example	<set id='PolygonMode'>1</set>	
Arguments	mode	Polygon Raster Mode
	Value range	0 = Normal mode – Gated triggering 1 = Continuous mode – Continuous triggering with selective gating

PolygonSync		
Description	Sets scan delay parameters and sync with SMC	
Syntax	<set id='PolygonSync'>{float tScanDelayInSec}</set>	
Example	<set id='PolygonSync'>0.0035</set>	
Arguments	tScanDelayInSec	Set the delay, based on polygon speed, from when the Start of Scan pulse is received, to when pixels are emitted. This delay is a composite delay referred to as the Work Delay in the overall polygon timing diagram.
	Value range	Based on calculations involving entries from the polygon device configuration and image size/placement considerations.

RasterParams (Polygon)		
Description	Sets pulse period for polygon raster	
Syntax	<set id='RasterParams'>{U16 pulsePeriodInTicks}</set>	
Example	<set id='RasterParams'>45</set>	
Arguments	pulsePeriodInTicks	Set pixel period

RasterParams (Polygon)		
	Value range	0 - 65535

LaserModDelay		
Description	Sets the modulation delay of the laser. A default value for this parameter can be set in the Laser Configuration file as parameter <u>LaserModDelay</u> .	
Syntax	<set id='LaserModDelay'>{U16 delay1InNsec},{U16 delay2InNsec}</set>	
Example	<set id='LaserModDelay'> 30,90 </set>	
Arguments	delay	The delay (in nano seconds) from the leading edge of LASER_MOD1 and LASER_MOD2 to the output of the first pulse on the LASER_GATE1 and LASER_GATE2 signals respectively.
	Value range	Within range of pulse period which is calculated by a device configuration file.

6.5.13 MARK-ON-THE-FLY SUPPORT

Marking on the fly (MOTF) support is provided through the use of several configuration and activation commands. Motion tracking in either the X or Y-axis can be configured using a digital quadrature input, or by simulating the motion in situations where an encoder feedback is not available but the motion speed is relatively constant.

The MOTF configuration is set using the parameters MotfCalFactor, MotfMode, and MotfDirection defined in the Controller Configuration file and additionally changeable as part of a job. Run-time control of the MOTF operation is performed through the use of the commands MotfEnable, MotfWaitForCount, MotfResetJump, MotfTrigger, and MotfWaitForTrigger. **Error! Reference source not found.** on page **Error! Bookmark not defined.** shows the intended use of these commands.

The actions of the MOTF commands are designed to permit multiple marking sequences within a single job, each of which requires separate frames of data that must be precisely spaced in distance. This normally occurs when the required markings exceed the physical limits of the lens field. Wire

marking applications are a good example of when different information must be marked at precise, but relatively long, distances along the length of the wire.

Mark-on-the-fly Parameters

MotfCalFactor		
Description	<p>Relates laser positioning bits to motion encoder counts. The default value for MotfCalFactor can be set as <u>MotfCalFactor</u> in the Controller Configuration file.</p> <p>Note: If used in a job, this command must appear after <set id='MotfDirection'>.</p>	
Syntax	<set id='MotfCalFactor'>{FLT calFactor}</set>	
Example	<set id='MotfCalFactor'> 23.345 </set>	
Arguments	calFactor	Calibration factor (in bit counts) for relating laser positioning bits to motion encoder counts. A negative number corresponds to a downward counting encoder tracking forward motion.
	Value range	-32768.0 - 32767.0

MotfDelayComp		
Description	<p>Sets run-time compensation for fixed reaction delays in the hardware from the time a <u>MotfWaitForCount</u> is executed to when marking actually occurs. This fixed time delay can result in variable positional offsets as a function of the speed of the material transport system.</p>	
Syntax	<set id='MotfDelayComp'>{U16 delay}</set>	
Example	<set id='MotfDelayComp'> 200 </set>	
Arguments	delay	Run-time compensation (in μ secs) for the fixed reaction delays in the hardware
	Value range	0 - 65535

MotfDirection		
Description	<p>MOTF orientation and direction in degrees. A default value for <u>MotfDirection</u> can be set in the Controller Configuration file.</p> <p>NOTE: This command must appear before <code><set id='MotfCalFactor'></code> and <code><set id='MotfMode'></code>.</p>	
Syntax	<code><set id='MotfDirection'>{U16 direction}</set></code>	
Example	<code><set id='MotfDirection'>270</set></code>	
Arguments	direction	Target travel direction relative to a galvo coordinate system.
	Value range	0 - left to right in the X axis 90 - bottom to top in the Y axis 180 - right to left in the X axis 270 - Top to bottom in the Y axis

MotfMode		
Description	<p>Defines how MOTF position information is derived. If an encoder option is selected, the quadrature encoder inputs are used. If a simulate-encoder option is selected, a 1Mhz clock is used to increment the encoder counter. A default value for <u>MotfMode</u> can be set in the Controller Configuration file.</p>	
Syntax	<code><set id='MotfMode'>{U16 mode}</set></code>	
Example	<code><set id='MotfMode'>0</set></code>	
Arguments	mode	Position tracking mode
	Value range	0 = Use encoder, 1D 1 = Simulate encoder, 1D 2 = Use encoders, 2D 3 = Simulate encoders, 2D

MotfTriggerEx		
Description	<p>This command performs the same function as MotfTrigger (see above), but it but adds an argument to specify a distance threshold that must be exceeded before the trigger logic is armed. When this command is issued—and immediately after a MotfWaitForTrigger command releases—the trigger counter is cleared and then counts until the threshold distance is exceeded. When the trigger distance is exceeded, the signal trigger logic is armed to look for the external trigger event. When the trigger event is seen, the counter is cleared once again and the MotfWaitForTrigger command will be armed for release when the count exceeds the specified value.</p>	
Syntax	<set id='MotfTriggerEx'>{U16 pin; U16 edge; U32 threshold}</set>	
Example	<set id='MotfTriggerEx'> 0; 1; 25000 </set>	
Arguments	pin	Input pin identifier
	Value range	0 – 31 Note: This must be the same input pin identifier as in WaitForIO .
	edge	The edge to trigger the start of the counter Note: The edge sense is dependent on how the input is wired.
	Value range	0 = Falling edge 1 = Rising edge
	threshold	Number of scaled encoder counts (in bits) to wait for before arming the trigger logic
	Value range	0 – $2^{32}-1$

MotfTriggerEvent		
Description	<p>This command is used to measure the distance an external transport system has traveled between transitions of an external signal. It is used in conjunction with the Priority message SetDigitalInputConfig. When executed, it enables the trigger counter for continuous counting and clears it. The trigger counter counts encoder counts in parallel with the normal MOTF operation and is not affected by normal MOTF state transitions.</p> <p>The command specifies an input pin that is expected be armed for event generation using the SetDigitalInputConfig message. If the specified signal causes a DigitalIO event, then the event message will contain the trigger couner value at the time of the event generation. The counter is then cleared and begins counting again.</p> <p>Note: The use of this command overrides the MotfTrigger and MotfTriggerEx commands..</p>	
Syntax	<set id='MotfTriggerEvent'>{U16 pin}</set>	
Example	<set id='MotfTriggerEvent'>2</set>	
Arguments	pin	Input pin identifier
	Value range	0 – 31 Note: The input pin identifier is the same as used in WaitForIO .

Mark-on-the-fly Commands

MotfEnable		
Description	<p>Enables or disables Mark-on-the-fly (MOTF) tracking.</p> <p>Upon enabling, the scaled MOTF encoder counts are added to the uVector values on each Jump and Mark vector. If in simulate mode (see MotfMode), the counter is incremented at a 1Mhz rate.</p> <p>Disabling does the following:</p> <ul style="list-style-type: none"> Disables uVector compensation Clears the HW encoder counter Zeros a firmware snapshot of the scaled HW counter Enables the HW encoder counter to count 	
Syntax	<MotfEnable>{U16 state}</MotfEnable>	
Example	<MotfEnable>1</MotfEnable>	
Arguments	state	Specifies whether MOTF tracking is to be enabled or disabled
	Value range	<p>0 = MOTF tracking is disabled.</p> <p>1 = MOTF tracking is enabled. Tracking happens only during Mark or Jump operations. Otherwise the galvos are held stationary.</p> <p>2 = MOTF tracking is enabled for continuous tracking. Tracking is immediate and galvos track the counters continuously except for during a MotfWaitForCount operation.</p> <p>3 = MOTF tracking is enabled for continuous tracking with edge of field detection. Tracking is immediate and galvos track the counters continuously except for during a MotfWaitForCount operation. If the galvos reach the edge of field while marking, the marking motion is temporarily suspended with the laser left on. Motion continues when the Motf counter biased position commands bring the galvos back into the field of view.</p>

MotfWaitForTrigger		
Description	<p>Configures MOTF to wait for the raw (unscaled) hardware encoder trigger counter to reach or exceed a specific value. (The trigger counter should have previously been armed using the <u>MotfTrigger</u> command.)</p> <p>The semantics are as follows:</p> <pre>while(abs (current encoder trigger counter)) < count)) wait; reset trigger condition and current encoder counter to zero.</pre>	
Syntax	<MotfWaitForTrigger>{U32 count}</MotfWaitForTrigger>	
Example	<MotfWaitForTrigger> 24557 </MotfWaitForTrigger>	
Arguments	count	Raw encoder count (in bits)
	Value range	0 - $2^{32}-1$

MotfResetJump		
Description	<p>Pre-positions the galvos to the start of the next field of patterns to be processed and takes into account the fact that the galvo starting points are not at the last "ideal" commanded position, but at a position offset by the MOTF counter.</p> <p>Note: At the time this command is executed, a snapshot of the MOTF counters is taken for possible use if the mode of the next <u>MotfWaitForCount</u> is "relative."</p>	
Syntax	<MotfResetJump{FLT xCoordinate; FLT yCoordinate; FLT zCoordinate; U16 jumpDelay}</MotfResetJump>	
Example	<MotfResetJump> -23000; 400; 0; 200 </MotfResetJump>	
Arguments	xCoordinate	Value that represents the X-axis coordinate of the start of the next field of patterns to be processed
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the X virtual field size of the system in floating point notation.

MotfResetJump		
	yCoordinate	Value that represents the Y-axis coordinate of the start of the next field of patterns to be processed
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Y virtual field size of the system in floating point notation.
	zCoordinate	Value that represents the Z-axis coordinate of the start of the next field of patterns to be processed
	Value range	$-2^{31} - 2^{31}-1$ Note: Depending on the unit selection, this value can take on a range which represents the Z virtual field size of the system in floating point notation.
	jumpDelay	Length of time (in μsecs) to delay after the execution of this command
	Value range	0 - 65535

LaserScribe		
Description	Turns the laser on for the specified number of MOTF counts.	
Syntax	<LaserScribe>{U32 count}</LaserScribe>	
Example	<LaserScribe> 2000 </LaserScribe>	
Arguments	count	Scaled encoder counts (in bits)
	Value range	0 - $2^{31}-1$

LaserRegulation	
Description	Conditions the system to dynamically adjust the laser pulse duty-cycle as a function of the speed of the material transport system.

LaserRegulation		
Syntax	<LaserRegulation>{U16 fMin; U16 fMax; FLT dcMin; FLT dcMax}</LaserRegulation>	
Example	<LaserRegulation> 2000; 10000; .3; .8 </LaserRegulation>	
Arguments	fMin	Minimum encoder frequency below which the laser duty-cycle is set to dcMin
	Value range	0 – 10000 encoder counts per 10ms
	fMax	Maximum encoder frequency above which the laser duty-cycle is set to dcMax
	Value range	0 – 10000 encoder counts per 10ms
	dcMin	Minimum duty-cycle expressed as a fraction
	Value range	0.0 – 1.0
	dcMax	Maximum duty-cycle expressed as a fraction
	Value range	0.0 – 1.0

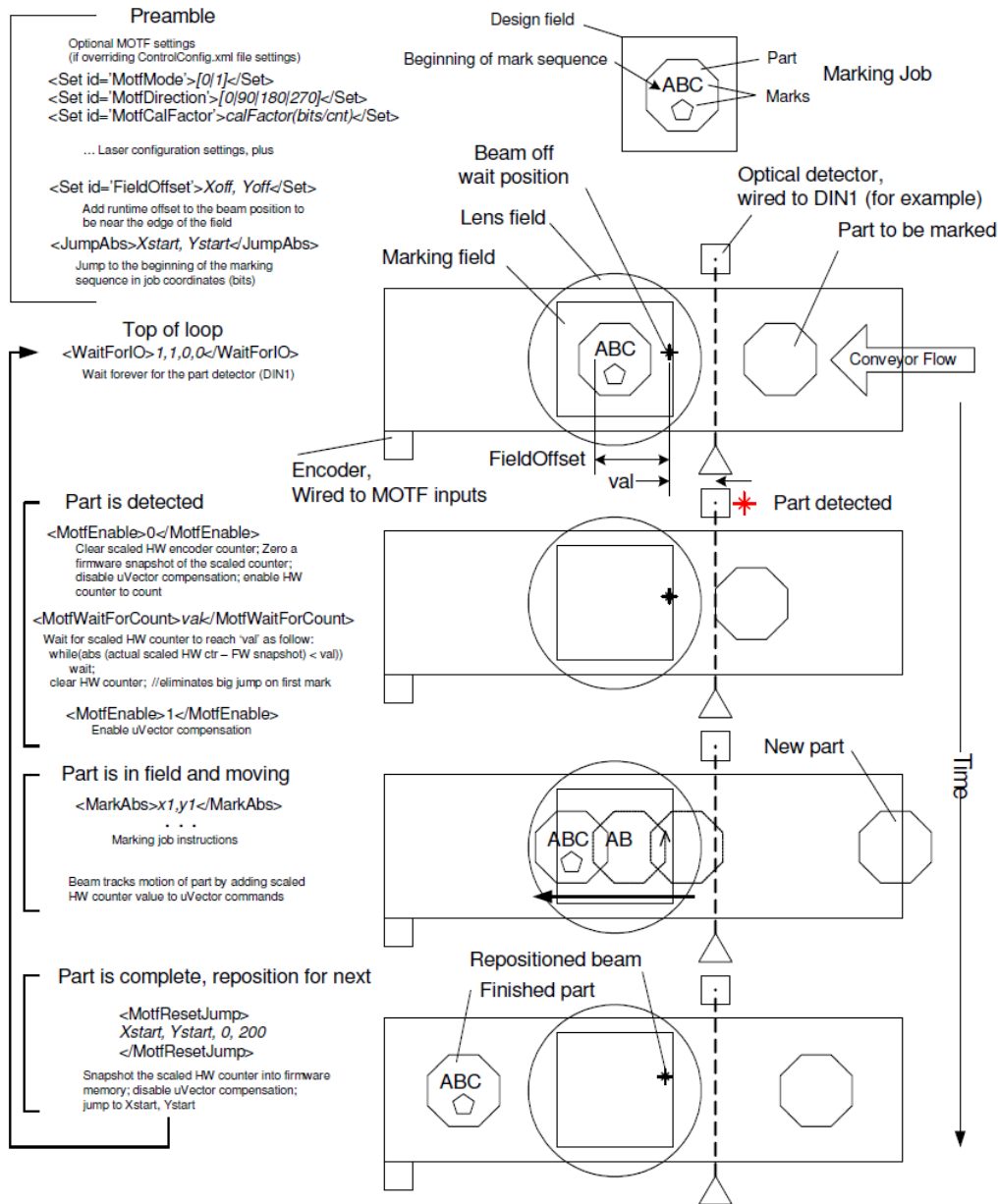


Figure 8 - MARK-ON-THY-FLY BASIC PROCESS FLOW

Instructions making up the MOTF loop can be sent to the SMC in advance of them being required as long as the job data does not vary. Synchronization with the external detectors is handled completely in the SMC.

SMC MOTF for fixed relative spacing of multiple fields (wire marking)

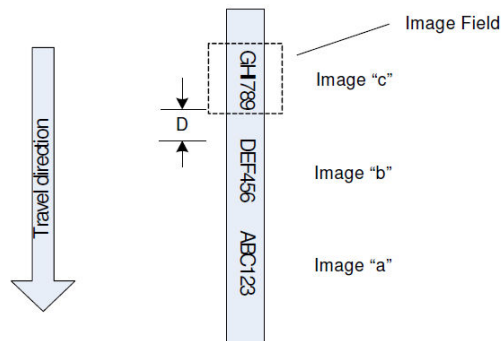


Figure 9 - MARK-ON-THE-FLY USAGE IN WIRE MARKING

Basic Job Structure**First time initialization:**

<!-- Activate MOTF counter logic. The MotfMode, MotfDirection, and MotfCalFactor parameters should be set prior to this point. -->

<MotfEnable>0</MotfEnable>

Job body:

<!-- A. Reset the MOTF Counter to zero and then enable counting -->

<MotfEnable>0</MotfEnable>

<!-- B. Wait for a fixed relative displacement from where we were when the MotfResetJump occurred -->

<MotfWaitForCount mode='relative'>distance-D</MotfWaitForCount>

<!-- C. Enable tracking -->

<MotfEnable>1</MotfEnable>

<!-- D. Pattern the first image "ABC123" while tracking -->

<JumpAbs>Xa0; Ya0; Za0</JumpAbs>

<MarkAbs>Xa1; Ya1; Za1</MarkAbs>

<MarkAbs>Xa2; Ya2; Za2</MarkAbs>

...

<!-- E. Disable tracking and jump to the beginning of the second "DEF456" vector set -->

<MotfResetJump>Xb0; Yb0; Zb0; 0</MotfResetJump>

<!-- F. Wait for a fixed relative displacement from where we were when the MotfResetJump occurred -->

<MotfWaitForCount mode='relative'>distance-D</MotfWaitForCount>

<!-- G. Enable tracking -->

<MotfEnable>1</MotfEnable>

<!-- H. Pattern the second image "DEF456" while tracking -->

<JumpAbs>Xb0; Yb0; Zb0</JumpAbs>

<MarkAbs>Xb1; Yb1; Zb1</MarkAbs>

<MarkAbs>Xb2; Yb2; Zb2</MarkAbs>

...

<!-- I. Disable tracking and jump to the beginning of the third "HIJ789" vector set -->

<MotfResetJump>Xc0; Yc0; Zc0; 0</MotfResetJump>

<!-- J. Wait for a fixed relative displacement from where we were when the MotfResetJump occurred -->

<MotfWaitForCount mode='relative'>distance-D</MotfWaitForCount>

<!-- K. Enable tracking -->

<MotfEnable>1</MotfEnable>

<!-- L. Pattern the third image "HIJ789" while tracking -->

<JumpAbs>Xc0; Yc0; Zc0</JumpAbs>

<MarkAbs>Xc1; Yc1; Zc1</MarkAbs>

<MarkAbs>Xc2; Yc2; Zc2</MarkAbs>

...

<!-- M. Disable tracking and jump to the beginning of the vector set -->

<MotfResetJump>Xa0; Ya0; Za0; 0</MotfResetJump>

<!-- N. Repeat steps A-M

SMC MOTF for multi-field imaging using 32-bit virtual addressing

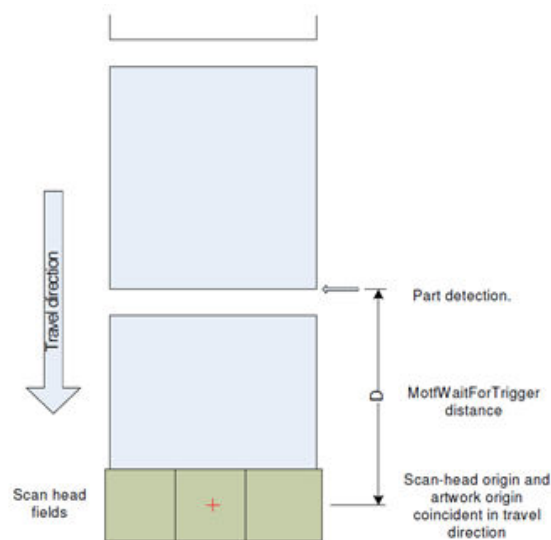


Figure 10 - MARK-ON-THE-FLY USAGE IN MULTI-IMAGE-FIELD APPLICATIONS

Basic job structure for each SMC controlling a scan-head

First time initialization:

<!-- Activate MOTF counter logic. Prior to this point, the MotfMode, MotfDirection, and MotfCalFactor parameters should be appropriately set. MotfEnable(0) initializes the logic and counters, begins counting, but does not permit galvo tracking of the MOTF motion -->

```
<MotfEnable>0</MotfEnable>
```

<!-- Set condition for MOTF HW trigger (e.g. START at logic level 1) Distance counter is reset to zero. If trigger condition is satisfied at the time of the command, the counter starts counting immediately. Otherwise, the counter starts counting on a transition from 0 to 1 of START. Note that any input signal can be selected per the same definition as WaitForIO -->

```
<set id='MotfTrigger'>0, 1</set>
```

Job body:

<!-- A. Wait for the trigger to be satisfied and then the count to be met or exceeded. The distance is expressed in scaled encoder counts as are used in the MotfWaitForCount instruction. The count should

be chosen such that travel-direction, origin of the artwork and the origin of the scan-head are coincident -->

<MotfWaitForTrigger>distance-D-in-unscaled-encoder-counts</MotfWaitForTrigger>

<!-- When the wait completes, the hardware trigger is automatically reset and monitoring of the next part is initiated -->

<!-- B. At this point the coordinate systems of the artwork and the galvo system need to be synchronized. This happens by resetting the MOTF logic again. The master scaled MOTF encoder counter resets to zero, but tracking is still disabled. -->

<MotfEnable>0</MotfEnable>

<!-- C. Now we wait until the vector set is in the field of view of the scan-head. This is expressed as dist-1 and is in artwork coordinates scaled to 32-bit virtual galvo comm and bits. The mode='absolute' attribute indicates that the wait is to use absolute scaled encoder counts. The normal (mode='relative') behavior is to wait for a count relative to the position when the last MotfResetJump occurred -->

<MotfWaitForCount mode='absolute'>dist-1</MotfWaitForCount>

<!-- D. Now we can mark the vectors but we must enable tracking first. The MotfEnable(1) command samples the current MOTF counter value which is then used in subsequent galvo motion commands to translate the artwork coordinates into the scan-head coordinates -->

<MotfEnable>1</MotfEnable>

<!-- E. Mark the vectors. Vector end points are specified in the artwork coordinate system scaled to 32-bit virtual galvo comm and bit units. These 32-bit virtual coordinates are translated into the scan-head command range by subtracting in real-time the MOTF encoder counter value sampled at the MotfEnable(1) command and the constantly incrementing MOTF encoder counter value -->

<JumpAbsEx>X0; Y0; Z0</JumpAbsEx>

<MarkAbsEx>X1; Y1; Z1</MarkAbsEx>

...

<!-- F. When all of the vectors in the current field have been imaged we disable tracking (but not counting) and re-position the galvos to a wait location that will minimize startup motion for the next vector set. The MOTF counter continues to increment thus tracking the material through the system. -->

<MotfResetJump>Xw; Yw; Zw</MotfResetJump>

<!-- G. Repeat sequence C-F for each frame that needs to be imaged. At the end of the entire job, begin at step A again. Since the trigger logic was automatically reset in the previous iteration it will have been triggered already and the counter will be very near the desired terminal value and step A will execute very quickly -->

6.5.14 VELOCITY CONTROLLED LASER MODULATION

Galvos make abrupt turns when rendering polygons, and the actual point of laser focus does not follow the ideal path described by the vectors. This is because of limitations of servo bandwidth imposed by finite inertia of the motors and mirrors, and restricted power supply voltage and current. Instead, the galvos follows a curved path joining one line segment to the next. These arcs introduce localized distortion of the final image, which is generally undesirable.

The PolyDelay parameter compensates for this effect by introducing a delay in the command stream generation. The delay gives the galvos time to reach the target destination before a new command directs them along the next vector segment. Normally the amount of time required reach the target destination is proportional to the angular change of the vector segments. Smaller angles require less time, and larger angles require more time. This proportionality is automatically managed using the VariPolyDelayFlag parameter.

The net effect of using non-zero PolyDelay values is that the laser focus point velocity slows down proportional to the length of the delay. Although the rendered geometry is more accurate, the energy density along the focus path increases in the regions of lower velocity. The same effect is also present at the beginning and end of marking vectors. A user manages these effects using the LaserOnDelay and LaserOffDelay parameters, which are normally adjusted to avoid “burn-in” effects at these points in a vector object.

The SMC provides a mechanism to automatically compensate for the effects of the changes in vector speed at the terminal and way-points of the vector drawing process. This mechanism offers three separate compensation modes, which would be selected based on the type of laser being used. An overview of this behavior is shown in the following figure.

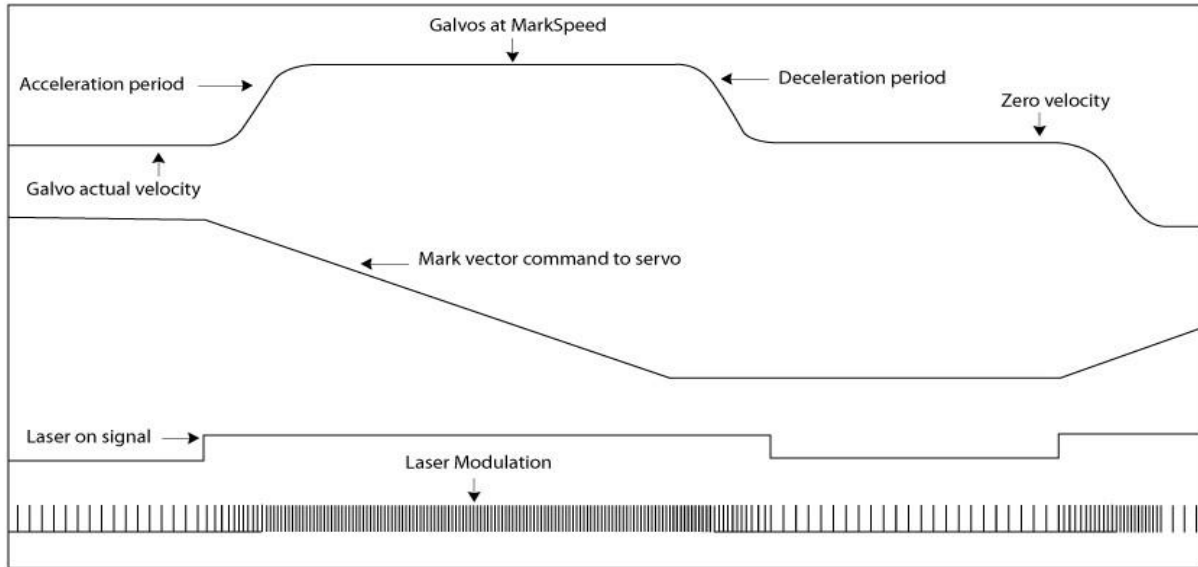


Figure 11 - VELOCITY CONTROLLED LASER MODULATION OVERVIEW

Mode 1 – Duty-cycle

Users of CO₂ lasers normally control average power by selecting an appropriate modulation duty-cycle. These lasers usually operate at a fixed frequency or pulse period, and the job varies the power by changing the pulse width. Mode 1 permits dynamic scaling of the pulse width from the normal job setting down to a settable percentage value of maximum power. This is illustrated in the following two figures, where the duty-cycle is varied between 80% and 20%.

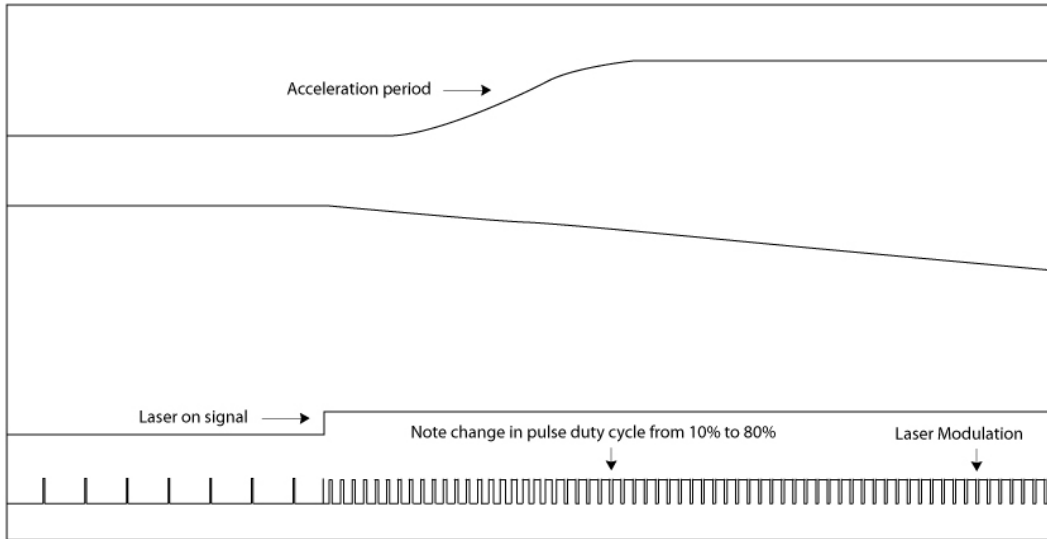


Figure 12 - VELOCITY CONTROLLED LASER MODULATION: DUTY-CYCLE, ACCELERATION EFFECT

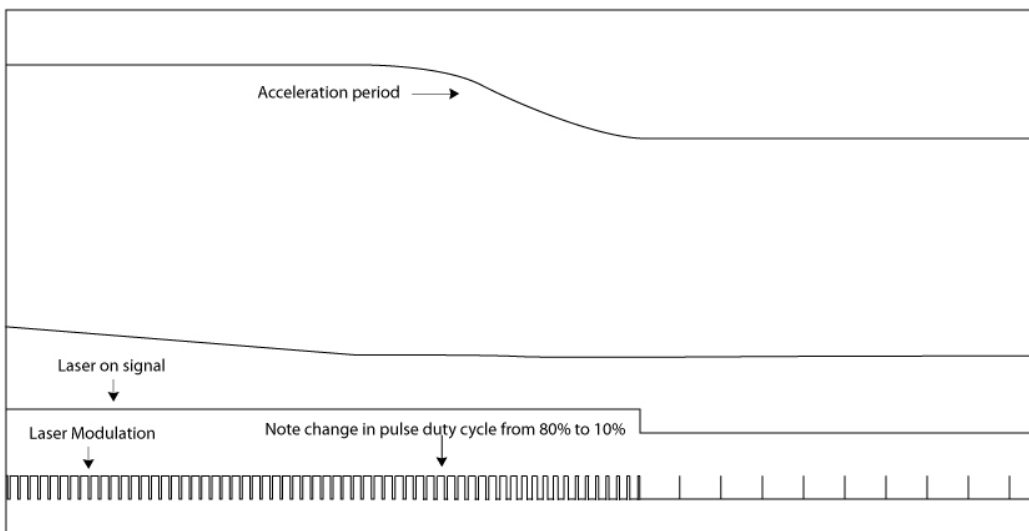


Figure 13 - VELOCITY CONTROLLED LASER MODULATION: DUTY-CYCLE, DECELERATION EFFECT

Mode 2 – Frequency

Users of YAG lasers have a choice of two power control modes. Since YAG lasers emit energy when they are Q-Switched, the individual pulse energy level can normally be controlled by changing the pumping energy and/or the modulation frequency. Mode 2 permits changing the average power by dynamically changing the pulse frequency while maintaining a constant pulse width. The frequency is reduced proportional to the galvo vector speed. This is illustrated in the following two diagrams which show the frequency changing from 100KHz down to 10KHz.

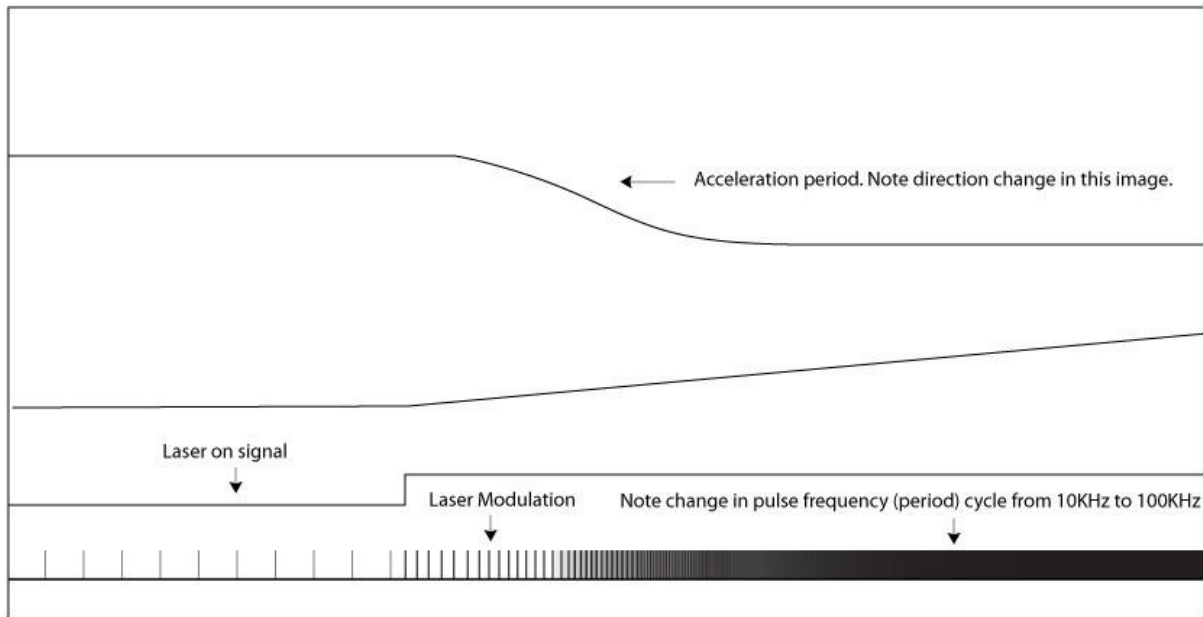


Figure 14 - VELOCITY CONTROLLED LASER MODULATION: FREQUENCY, ACCELERATION EFFECT

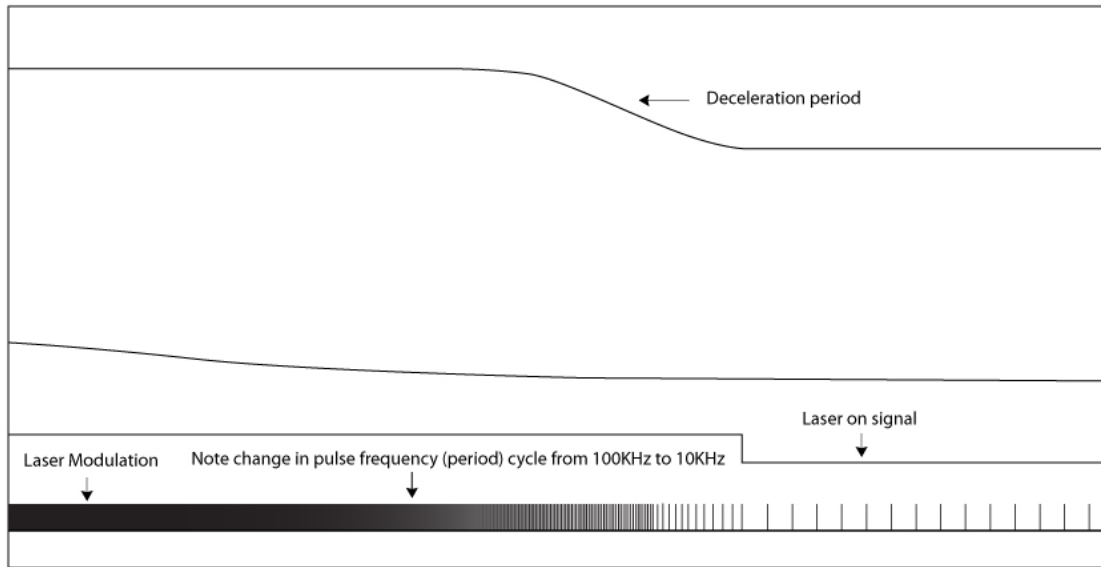


Figure 15 - VELOCITY CONTROLLED LASER MODULATION: FREQUENCY, DECELERATION EFFECT

Mode 3 – Laser Power

Mode 3 controls the analog or digital laser power setting proportional to the velocity. This is illustrated below where laser power, represented by an analog control voltage, varies between 80 and 20%. Note that the laser modulation does not change in this mode.

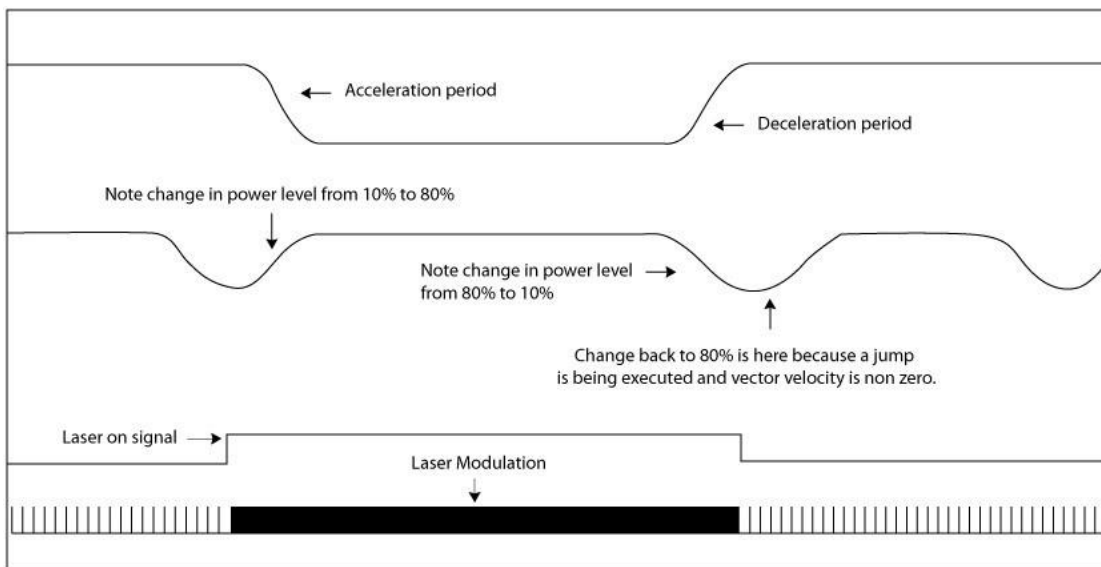


Figure 16 - VELOCITY CONTROLLED LASER MODULATION: LASER POWER

Velocity Controlled Laser Modulation Compensation

The VelocityComp command is used to implement velocity controlled laser modulation compensation.

VelocityComp		
Description	Sets the mode and behavior of the velocity controlled laser modulation compensation. If the first argument is zero, then arguments two and three may be omitted.	
Syntax	<set id='VelocityComp'>{U16 mode; U16 maxComp; U16 aggressivity}</set>	
Example	<set id='VelocityComp'> 1; 20; 1000 </set>	
Arguments	mode	Mode of operation for velocity-controlled laser modulation compensation
	Value range	0 = Disabled 1 = Duty-cycle (pulse width changes) 2 = Frequency (pulse period changes) 3 = Power (analog or digital power changes)
	maxComp	The limit of the compensation that will be done in terms of percentage of the maximum possible value. Compensation will be applied proportional to the calculated vector velocity of the mirror.
	Value range	0 - 100
	aggressivity	How aggressively the system will compensate for velocity changes. The higher the number, the quicker the change will be applied. This number is in Hertz, and it directly correlates to the tuning bandwidth of the galvo servos.

VelocityComp		
	Value range	500 - 5000

6.5.15 VIA-HOLE DRILLING SUPPORT

The SMC has several extensions designed to support open-loop and closed-loop laser drilling modes. These extensions will work with galvo/servo systems that provide real-time in-position feedback via Digital I/O, XY2-100 Status, or GSBUS Status.

Drilling data can be applied using two different commands:

1. The JumpAndFireList command which specifies a list of discrete two- or three-axis coordinate data along with laser pulse-width values for two laser modulation outputs.
2. The JumpAndDrillList command which specifies a list of discrete two-axis coordinate data with drilling specific laser firing and synchronization parameters.

The coordinate information in both of these commands represents discrete jump points that are applied without profiling. Galvo/servo controllers used in this mode must be capable of handling transient command inputs that could range in distance from single-bit to full-field. The expectation is that real-time in-position feedback is available for sensing by the SMC as is the case with GSBUS connected Lightning™ II digital servo controllers. Both closed-loop and open-loop modes of operation are supported as described below. More detailed information about via-hole drilling using the SMC can be found in application notes on the Cambridge Technology web site www.camtech.com/downloads/customers. Please contact Cambridge Technology Technical Support for the download password: support-us@cambridgetechnology.com

Closed-loop operation

In fully closed-loop mode, the laser firing part of the JumpAndFireList and JumpAndDrillList execution is configured to check for in-position before firing. Checking for up to four axes is done in parallel. A programmable timeout is used to protect against abnormal settling times or galvo fault conditions. In such a case, drilling is stopped and an exception event is generated and forwarded to the host application for handling.

SettleCheckMode is used to configure closed loop drilling behavior.

SettleCheckMode		
Description	Sets the settle-checking behavior of the JumpAndFireList and JumpAndDrillList commands. Used to validate the position of the galvos after a move is made and before the laser is fired.	
Syntax	<set id='SettleCheckMode'>{U16 input; HEX U32 mask; HEX U32 value; U16 timeout; U16 checkDelay; U16 checkMode }</set>	
Example	<set id='SettleCheckMode'>3; 0x00006666; 0x00006666; 10000; 80; 0</set>	
Arguments	input	Selects the settle-checking inputs.
	Value range	0 = disabled 1 = check XY2-100 status 2 = check standard digital I/O 3 = check GSBUS status
	mask	Bits to consider (hex) in 32-bit units
	Value range	0 - 0xFFFFFFFF
	value	Bit values when settled (hex) in 32-bit units
	Value range	0 - 0xFFFFFFFF
	Timeout / FiringAdjust	If <i>checkMode</i> = 0, <i>timeout</i> defines how long to wait (in usec) for <i>value</i> to match the <i>mask</i> . If the <i>timeout</i> value is exceeded, and exception is generated and the job aborted. If <i>checkMode</i> = 1 or 2, (semi-open-loop or full-open-loop), the settle checking is done after the greater of <i>settleCheckDelay</i> or the maximum jump time selected from the axis jump-time tables for the distance requested. In these cases, the <i>timeout</i> value is interpreted as a firing-adjust value and is added to the jump delay to programmatically <i>increase</i> or <i>decrease</i> the amount of time before firing the laser.

SettleCheckMode		
	Value range	When interpreted as a timeout: 0 - 85.899 sec When interpreted as a firing-adjust: -500 to 500 usec
	checkDelay	How long to wait (in μ secs) before checking for settling after initiating a jump. This provides time for the galvos to go out of position before they are checked for arrival at the new position. If <i>mode</i> = 1, both the XY2-100 and XY2-100e status ports are examined concurrently. XY2-100 status bits are in position[15..0] and XY2-100e are bits position[31..16]. If <i>mode</i> = 2, the bits are interpreted as defined in the <u>CurrentDIO</u> value of the Broadcast Status Data packet. If <i>mode</i> = 3, the GSBUS status register is compared where four-bit fields are used for each axis. Note: If <i>checkMode</i> does not = 0, then this value acts as a minimum jump time specification overriding the data calculated during the <u>CalibrateJumpTime</u> operation.
	Value range	0 - 85.899 sec
	checkMode	Selects the checking behavior.
	Value range	0 = Before firing the laser (closed-loop mode) 1 = After firing the laser (semi-open-loop, uses jump-time table) 2 = Do not check (full open-loop, uses jump-time table) Note: If <i>checkMode</i> is set to 1, in position checking is performed <u>after</u> the galvos have been commanded to move. Therefore there is only a short interval of time when in-position may still be valid.

Open-loop operation

In open-loop mode drilling the galvos are calibrated for the amount of time it takes to execute a jump and reach an in-position condition. During calibration, a sequence of variable length jumps is executed and the settling time recorded in a table, one table for each axis. During execution of the `JumpAndFireList` or `JumpAndDrillList` command, the distance required of each axis is used to index each of the tables and jump times retrieved. If the distance does not fall on a table entry, then linear interpolation between table entries is performed to calculate a value. The maximum of the table values retrieved is used to wait before firing. In this mode it is possible to cause the laser to fire earlier or later using `FiringAdjust` parameter. Firing earlier may permit improved throughput at the sacrifice of some quality.

The galvos are commanded to jump to the next as soon as the laser firing starts. This permits overlap of operations recognizing the fact that galvo inertia prevents instantaneous motion when a command is received.

Calibration of the jump-times is invoked using the `CalibrateJumpTime` command. This command is available for use only with Lightning II galvos systems connected to the SMC via the GSBUS. The command `SettleCheckMode` must be used prior to `CalibrateJumpTime` to set the properties describing how settle checking is to be performed during calibration.

CalibrateJumpTime		
Description	Builds run-time tables of measured jump times which are then used in the execution of the <code>JumpAndFireList</code> or <code>JumpAndDrillList</code> commands. The resulting table data is used to calculate the time that the galvos will achieve in-position status based on a requested jump distance.	
Syntax	<CalibrateJumpTime>{HEX U32 axisMask; U32 averagingMode; FLT maxDistance; FLT smallestStep; BOOL logData }</CalibrateJumpTime>	
Example	<CalibrateJumpTime>0x3, 0, 80.000000, 0.100000, TRUE</CalibrateJumpTime>	
Arguments	axisMask	Select the X and Y axes of a head for calibration. Two-bit-per-head bit-mask. The least significant of the two bit field is the X axis. Multiple heads are specified by enabling additional two-bit fields in successively higher-order bit fields.

CalibrateJumpTime		
	Value range	0x0 = No axes selected 0x1 = Head 1, X axis 0x2 = Head 1, Y axis 0x3 = Head 1, X and Y axis 0x4 = Head 2, X axis 0x8 = Head 2, Y axis 0xC = Head 2, X and Y axis Other axis combinations are specified by logical OR-ing the bit-fields together.
	averagingMode	How the measured data is averaged. Multiple samples for a given distance are taken in various parts of the field.
	Value range	0 – the table value gets the average of the samples for a given distance 1 – the table value gets the maximum of the samples for a given distance 2 – the table value gets the minimum of the samples for a given distance
	maxDistance	Specifies the largest distance to calibrate. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.
	Value range	1 – $(2^{24}-1)$ (bits) or 1 field size (user units)
	smallestStep	Specifies the smallest distance to calibrate. Values are floating point and are converted into system “bits” units per the <u>Units</u> command.

CalibrateJumpTime		
	Value range	1 – (2 ²⁴ -1) (bits) or 1 field size (user units) Note: <i>smallestStep</i> must be less than <i>maxDistance</i>
	logData	Specifies if a log file is created to hold the calibration data. This file is created inside the SMC and can only be accessed under the guidance of Cambridge Technology technical support personnel. It is used for diagnostic purposes only.
	Value range	TRUE or FALSE. FALSE is recommended for normal operation.

Binary interface for JumpAndFireList data

The JumpAndFireList XML command can pass up to 65536 discrete drill points in a single instruction. This can be a large amount of ASCII data when represented in XML format and can be inefficient to generate with certain compilers. The API provided special binary interfaces to pass the JumpAndFireList data without converting to XML. Each call to these methods creates a job packet that is sent to the SMC for execution just as if it were passed as XML. Both 2-D and 3-D methods are exposed using streaming or structured job control.

sendJumpAndFireList2D	
Purpose	Sends binary 2-D JumpAddFireList streaming data to an SMC device session

sendJumpAndFireList2D		
Syntax	Uint sendJumpAndFireList2D (ushort numPoints, float[] xCoord, float[] yCoord, uint[] laserValue, ushort outputMode, ushort laserOnDelay, ushort laserOnTime)	
Arguments	numPoints	The number of data points in the list
	xCoord	An array of X coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.
	yCoord	An array of X coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.
	laserValue	An array of laser values that are applied per the <i>outputMode</i> setting. The array length is expected to be <i>numPoints</i> long.
	outputMode	Specifies how to interpret <i>laserValue[n]</i> : 0 = Interpret <i>laserValue[n]</i> as a laser pulse-width pair (laser-ticks) 1 = Interpret <i>laserValue[n]</i> as Analog Port 1 value (12-bits) 2 = Interpret <i>laserValue[n]</i> as Analog Port 2 value (12-bits) 3 = Interpret <i>laserValue[n]</i> as Digital power port value (8-bits)
	laserOnDelay	Specifies the waiting period (in μ secs) before firing after an incremental jump.

sendJumpAndFireList2D		
	laserOnTime	Specifies the duration that the laser is fired (in laser ticks).
Comments	<p>The value range of <i>laserValue[n]</i> is mode-dependent.</p> <p>For <i>outputMode</i> = 0, the value represents individual pulse width settings in laser ticks for LASER_MOD1 and LASER_MOD2. The LASER_MOD1 setting is specified in bits [15 – 0], and the LASER_MOD2 setting in bits [31 – 16]</p> <p>For other settings of outputMode, the value is contained in the least-significant 16-bits of the value.</p>	
See also	JumpAndFireList	

sendJumpAndFireList3D		
Purpose	Sends binary 3-D JumpAddFireList streaming data to an SMC device session	
Syntax	Uint sendJumpAndFireList3D (<div> ushort numPoints, float[] xCoord, float[] yCoord, float[] zCoord, uint[] laserValue, ushort outputMode, ushort laserOnDelay, ushort laserOnTime) </div>	
Arguments	numPoints	The number of data points in the list
	xCoord	An array of X coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.

sendJumpAndFireList3D		
	yCoord	An array of X coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.
	zCoord	An array of Z coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.
	laserValue	An array of laser values that are applied per the <i>outputMode</i> setting. The array length is expected to be <i>numPoints</i> long.
	outputMode	Specifies how to interpret <i>laserValue[n]</i> : 0 = Interpret <i>laserValue[n]</i> as a laser pulse-width pair (laser-ticks) 1 = Interpret <i>laserValue[n]</i> as Analog Port 1 value (12-bits) 2 = Interpret <i>laserValue[n]</i> as Analog Port 2 value (12-bits) 3 = Interpret <i>laserValue[n]</i> as Digital power port value (8-bits)
	laserOnDelay	Specifies the waiting period (in μ secs) before firing after an incremental jump.
	laserOnTime	Specifies the duration that the laser is fired (in laser ticks).
Comments	<p>The value range of <i>laserValue[n]</i> is mode-dependent.</p> <p>For <i>outputMode</i> = 0, the value represents individual pulse width settings in laser ticks for LASER_MOD1 and LASER_MOD2. The LASER_MOD1 setting is specified in bits [15 – 0], and the LASER_MOD2 setting in bits [31 – 16]</p> <p>For other settings of <i>outputMode</i>, the value is contained in the least-significant 16-bits of the value.</p>	

sendJumpAndFireList3D	
See also	JumpAndFireList

sendJumpAndFireList3DSegment		
Purpose	Sends binary 3-D JumpAddFireList as a deferred-execution named segment to an SMC device session	
Syntax	Uint sendJumpAndFireList3D (<div> ushort numPoints, float[] xCoord, float[] yCoord, float[] zCoord, uint[] laserValue, ushort outputMode, ushort laserOnDelay, ushort laserOnTime, string segmentId) </div>	
Arguments	numPoints	The number of data points in the list
	xCoord	An array of X coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.
	yCoord	An array of Y coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.
	zCoord	An array of Z coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.
	laserValue	An array of laser values that are applied per the <i>outputMode</i> setting. The array length is expected to be <i>numPoints</i> long.

sendJumpAndFireList3DSegment		
	outputMode	Specifies how to interpret <i>laserValue[n]</i> : 0 = Interpret <i>laserValue[n]</i> as a laser pulse-width pair (laser-ticks) 1 = Interpret <i>laserValue[n]</i> as Analog Port 1 value (12-bits) 2 = Interpret <i>laserValue[n]</i> as Analog Port 2 value (12-bits) 3 = Interpret <i>laserValue[n]</i> as Digital power port value (8-bits)
	laserOnDelay	Specifies the waiting period (in µsecs) before firing after an incremental jump.
	laserOnTime	Specifies the duration that the laser is fired (in laser ticks).
	segmentID	Specifies the name of the segment
Comments	<p>The value range of <i>laserValue[n]</i> is mode-dependent.</p> <p>For <i>outputMode</i> = 0, the value represents individual pulse width settings in laser ticks for LASER_MOD1 and LASER_MOD2. The LASER_MOD1 setting is specified in bits [15 – 0], and the LASER_MOD2 setting in bits [31 – 16]</p> <p>For other settings of outputMode, the value is contained in the least-significant 16-bits of the value.</p> <p>The list is packaged into the named deferred execution segment and sent to the SMC for later execution via a <u>Sequence</u> command.</p>	
See also	<u>JumpAndFireList, Structured Job Commands</u>	

Binary interface for JumpAndDrillList data

The JumpAndDrillList XML command can pass up to 65536 discrete drill points in a single instruction. This can be a large amount of ASCII data when represented in XML format and can be inefficient to generate with certain compilers. The API provide special binary interfaces to pass the

JumpAndDrillList data without converting to XML. Each call to these methods creates a job packet that is sent to the SMC for execution just as if it were passed as XML.

sendJumpAndDrillList		
Purpose	Sends binary JumpAddDrillList streaming data to an SMC device session	
Syntax	Uint sendJumpAndDrillList (<div> ushort numPoints, float[] xCoord, float[] yCoord, ushort laserOnTime, ushort laserFireMode, ushort extSyncPin, ushort extSyncPinState) </div>	
Arguments	numPoints	The number of data points in the list
	xCoord	An array of X coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.
	yCoord	An array of X coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.
	laserOnTime	The laser firing time in LaserTick units

sendJumpAndDrillList		
	laserFireMode	<p>Specifies what to do after firing the laser</p> <p>0 = Fire the laser and do not wait. Immediately jump to the next location.</p> <p>1 = Fire the laser and wait until it is on. This accommodates any <u>LaserOnDelay</u> that may be specified.</p> <p>2 = Fire the laser and wait until it is off. This accomodates the <u>LaserOnDelay</u>, <u>LaserOnTime</u>, and <u>LaserOffDelay</u></p> <p>3 = Fire the Laser and wait until an external signal specified by the optional argument <i>ExtSyncPin</i> is asserted to the state specified by the optional argument <i>ExtSyncPinState</i>.</p>
	extSyncPin	Specifies the external pin to sense. The pin identifier is the same as the portNumber argument in the command <u>WaitForIO</u>
	extSyncPinState	Specifies the logical state of the external pin being sensed. The state should take into consideration assertion inversions due to signal conditioning circuitry.
Comments	<i>extSyncPin</i> and <i>extSyncPinState</i> are interpreted only if the <i>laserFireMode</i> is set to 3. Set the values to zero if laserFireMode is 0 – 2.	
See also	<u>JumpAndFireList</u>	

sendJumpAndDrillListSegment	
Purpose	Sends binary JumpAddDrillList as a deferred-execution named segment to an SMC device session

sendJumpAndDrillListSegment		
Syntax	<pre> Uint sendJumpAndDrillListSegment (ushort numPoints, float[] xCoord, float[] yCoord, ushort laserOnTime, ushort laserFireMode, ushort extSyncPin, ushort extSyncPinState string segmentID) </pre>	
Arguments	numPoints	The number of data points in the list
	xCoord	An array of X coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.
	yCoord	An array of X coordinate values for the points in the list. The array length is expected to be <i>numPoints</i> long.
	laserOnTime	The laser firing time in LaserTick units
	laserFireMode	<p>Specifies what to do after firing the laser</p> <p>0 = Fire the laser and do not wait. Immediately jump to the next location.</p> <p>1 = Fire the laser and wait until it actually is on. This accommodates any <u>LaserOnDelay</u> that may be specified.</p> <p>2 = Fire the laser and wait until it is off. This accomodates the <u>LaserOnDelay</u>, <i>LaserOnTime</i>, and <u>LaserOffDelay</u></p> <p>3 = Fire the Laser and wait until an external signal specified by the optional argument <i>ExtSyncPin</i> is asserted to the state specified by the optional argument <i>ExtSyncPinState</i>.</p>

sendJumpAndDrillListSegment		
	extSyncPin	Specifies the external pin to sense. The pin identifier is the same as the portNumber argument in the command WaitForIO
	extSyncPinState	Specifies the logical state of the external pin being sensed. The state should take into consideration assertion inversions due to signal conditioning circuitry.
	segmentID	Specifies the name of the segment.
Comments	<i>extSyncPin</i> and <i>extSyncPinState</i> are interpreted only if the <i>laserFireMode</i> is set to 3. Set the values to zero if laserFireMode is 0 – 2.	
See also	JumpAndFireList	

6.6 STRUCTURED JOB ORGNIZATION

Any job data defined above, from single statement to a lengthy sequence of statements, can be passed to the SMC for immediate execution via the `sendStreamData` method. Data sent like this is executed once and then discarded. If a repetitive marking pattern is desired, an application could repeatedly send the job data with a sequence of calls to `sendStreamData`. Alternatively, jobs can be structured into groups of related statements called segments and these segments can be sent to the SMC as a named entity for deferred execution. Many segment definitions may be sent to the SMC in this manner. A separate sequence list can then be used to dictate the execution order of the segments, how many times to iterate each segment, and how many times to iterate the sequence as a whole.

An entire job made up of multiple segments, and potentially multiple sequences, can be sent in a single `sendStreamData` call. The same XML that makes up this job can be passed to the `saveJobData` method for storage on the SMC and later accessed in stand-alone operational mode. One or more segment definitions may be also specified and saved as a library for later reference and use within a sequence specification. This greatly reduces the amount of data moving through the system when commonly used graphical entities such as pre-rendered character sets are required at run-time.

6.6.1 SEGMENT CONSTRUCT

Segment		
Description	<p>Defines a job segment, which is a group of related instructions. Any job action command or parameter statement is valid inside of a Segment. Multiple Segments can be defined inside of a call to <u>sendStreamData</u>.</p> <p>Note: This command is valid only in a job <Data> definition.</p>	
Syntax	<pre><Segment id='{STR name}' iterations='{U16 iterations}' deferred='{BOOL deferred}'> {any valid series of command or parameter statements} </Segment></pre>	
Example	<pre><Segment id='LaserCfg' iterations='1' deferred='true'> <set id='LaserPulse'>1; 50; 100</set> <LaserPower>200</LaserPower> </Segment></pre>	
Arguments	name	A name assigned to this segment
	Value range	Up to 128 alphanumeric characters
	iterations	The number of times this segment is to be iterated. The default is 1 if not specified.
	Value range	1 - 65535
	deferred	Specifies whether the segment is executed immediately or is saved for reference by a Sequence. If no value is specified, the default is false (execute immediately).
	Value range	true = Save this segment for reference by a Sequence false = Execute this segment immediately

6.6.2 STRUCTURED JOB SEQUENCING

Sequence		
Description	Defines the sequence of execution for job segments that were previously defined with the Segment command. Note: This command is valid only in a job <Data> definition.	
Syntax	<Sequence iterations='{U16 iterations}'> {any valid series of sequence statements} </Sequence>	
Example	<Sequence iterations='3'> <RunSegment> LaserCfg </RunSegment> <RunSegment> Vectors; 5 </RunSegment> </Sequence>	
Arguments	iterations	The number of times to execute this job segment. A value of 0 means to execute this sequence continuously.
	Value range	0 - 65535

Sequence Commands

RunSegment		
Description	Causes a previously loaded and “deferred” job segment to be executed. Note: This command is valid only inside a <Sequence> definition.	
Syntax	<RunSegment>{STR segmentName; U16 iteration}</RunSegment>	
Example	<RunSegment> Vectors; 5 </RunSegment>	
Arguments	segmentName	Identifier of a previously loaded and “deferred” job segment
	Value range	Up to 128 alphanumeric characters

RunSegment		
	iteration	Number of times to iterate the named job segment. If the previously loaded job segment had an iteration attribute specified, then the two iteration values are multiplied, and the result is the final iteration count. If not specified, the default is 1.
	Value range	1 - 65535

DeleteSegment		
Description	Causes a previously loaded and “deferred” job segment to be discarded with all used memory returned to the main memory pool. Note: This command is valid only inside a <Sequence> definition.	
Syntax	<DeleteSegment>{STR segmentName}</DeleteSegment>	
Example	<DeleteSegment> LaserCfg </DeleteSegment>	
Arguments	segmentName	Identifier of a previously loaded and “deferred” job segment
	Value range	Up to 128 alphanumeric characters

DeleteAllSegments		
Description	Causes all previously loaded and “deferred” segments to be discarded with all used memory returned to the main memory pool.	
Syntax	<DeleteAllSegments></DeleteAllSegments>	
Example	<DeleteAllSegments></DeleteAllSegments>	
Arguments	None	

DisableSegment		
Description	Causes a previously loaded and “deferred” job segment to be marked as “disabled”, which causes it to be skipped when encountered within a subsequent sequence list. Note: This command is valid only inside a <Sequence> definition.	
Syntax	<DisableSegment>{STR segmentName}</DisableSegment>	
Example	<DisableSegment> LaserCfg </DisableSegment>	
Arguments	segmentName	Identifier of a previously loaded and “deferred” job segment
	Value range	Up to 128 alphanumeric characters

EnableSegment		
Description	Causes a previously loaded and “deferred” job segment to be marked as “enabled”, which causes it to be executed when encountered within a subsequent sequence list. Note: This command is valid only inside a <Sequence> definition.	
Syntax	<EnableSegment>{STR segmentName}</EnableSegment>	
Example	<EnableSegment> LaserCfg </EnableSegment>	
Arguments	segmentName	Identifier of a previously loaded and “deferred” job segment
	Value range	Up to 128 alphanumeric characters

UsingFile	
Description	<i>(Reserved for future use)</i> Specifies the name of a previously saved set of <Segment> definitions for use in a following <Sequence> definition. Note: This command is valid only in a job <Data> definition.
Syntax	<UsingFile>{STR segmentFileName}</UsingFile>
Example	<UsingFile> LaserSettings </UsingFile>

UsingFile		
Arguments	segmentFileName	Identifier of a previously saved set of <Segment> definitions. These definitions would have been saved to the SMC using the API method <u>savejobData</u> .
	Value range	Up to 128 alphanumeric characters

Note: Do not mix deferred and non-deferred segments in a single XML job packet.

6.6.3 STRUCTURED JOB EXAMPLE

Table 23 - STRUCTURED JOB EXAMPLE	
XML Job Statement	Meaning
<Data type='JobData' rev='2.0'>	Define a job data packet. API Action: Prepare a job packet
<Segment id='Preamble' iterations='1' deferred='TRUE'>	Define a deferred execution segment.
<BeginJob></BeginJob>	Assert BUSY signal and generate an event.
<set id='ActiveCorrectionTable'>1</set>	Select the marking laser correction table.
<set id='EnableLaser'>TRUE</set> </Segment>	Enable the laser for marking. Delimit the segment. API Action: Compile and mark for on-the-board in-memory staging; append to output buffer.
<Segment id='Alignment' deferred='TRUE'>	Define an immediate execution segment.

Table 23 - STRUCTURED JOB EXAMPLE	
XML Job Statement	Meaning
<code><set id='FieldOffset'>0.000000; 0.000000; 0.000000</set></code>	Introduce a field offset.
<code><set id='Transform'>1.000000; 0.000000; 0.000000; 1.000000</set></code>	Set a Unity transform.
<code></Segment></code>	Delimit the segment. API Action: Compile and mark for on-the-board in-memory staging; append to output buffer.
<code><Segment id='Params:Default' deferred='TRUE'></code>	Define a deferred execution segment.
<code><set id='LaserPower'>50</set></code> <code><set id='LaserEnableDelay'>15</set></code> <code><set id='LaserEnableTimeout'>15</set></code> <code><set id='LaserOnDelay'>0</set></code> <code><set id='LaserOffDelay'>50</set></code> <code><set id='LaserPipelineDelay'>100</set></code> <code><set id='LaserPulse'>1; 5; 10</set></code>	Set the laser parameters.
<code><set id='MarkSpeed'>10; 10</set></code> <code><set id='JumpSpeed'>10; 10</set></code>	Set the galvo speeds.
<code></Segment></code>	Delimit the segment. API Action: Compile and mark for on-the-board in-memory staging; append to output buffer.
<code><Segment id='Vectors:Pentagon.plt' iterations='1' deferred='TRUE'></code>	Define a deferred execution segment.
<code><set id='JumpDelay'>100</set></code>	Set the delays. These must be kept with the vector definitions.

Table 23 - STRUCTURED JOB EXAMPLE	
XML Job Statement	Meaning
<code><set id='MarkDelay'>100</set></code> <code><set id='PolyDelay'>50</set></code> <code><set id='VariPolyDelayFlag'>FALSE</set></code>	
<code><JumpAbs>-10000; 10000; 0</JumpAbs></code> <code><MarkAbs>0; 20000; 0</MarkAbs></code> <code><MarkAbs>10000; 10000; 0</MarkAbs></code> <code><MarkAbs>7500; -10000; 0</MarkAbs></code> <code><MarkAbs>-7500; -10000; 0</MarkAbs></code> <code><MarkAbs>-10000; 10000; 0</MarkAbs></code>	Perform marking operations.
<code></Segment></code>	Delimit the segment. API Action: Compile and mark for on-the-board in-memory staging; append to output buffer.
<code><Segment id='Postamble' iterations='1' deferred='TRUE'></code>	Define a deferred execution segment.
<code><set id='EnableLaser'>FALSE</set></code>	Enables the pointer laser.
<code><set id='ActiveCorrectionTable'>2</set></code>	Select the pointer laser correction table.
<code><EndJob></EndJob></code>	De-assert BUSY signal and generate an event.
<code></Segment></code>	Delimit the segment. API Action: Compile and mark for on-the-board in-memory staging; append to output buffer.
<code><Sequence iterations='1'></code>	Define a sequence to be iterated 1 time.
<code><RunSegment>Preamble</RunSegment></code>	Execute the preamble segment.
<code><RunSegment>Alignment</RunSegment></code>	Execute the alignment segment.

Table 23 - STRUCTURED JOB EXAMPLE	
XML Job Statement	Meaning
<pre><RunSegment>Params:Default</RunSegment> </Sequence></pre>	<p>Execute the params segment.</p> <p>End the sequence</p> <p>API Action: Compile and mark for on-the-board in-memory staging; append to output buffer.</p>

<pre><Sequence iterations='10'> <RunSegment>Vectors:Pentagon.plt</RunSegment> </Sequence></pre>	<p>Define a sequence to be iterated 10 times.</p> <p>Execute the marking vectors.</p> <p>End the sequence.</p> <p>API Action: Compile and mark for on-the-board in-memory staging; append to output buffer.</p>
<pre><Sequence iterations='1'> <RunSegment>Postamble</RunSegment> </Sequence></pre>	<p>Define a sequence to be iterated 1 time.</p> <p>Execute the postamble segment.</p> <p>End the sequence.</p> <p>API Action: Compile and mark for on-the-board in-memory staging; append to output buffer.</p>
<pre></Data></pre>	<p>End the job packet.</p> <p>API Action: Send output buffer to board.</p> <p>Controller Actions: Deferred segments are staged in memory on the controller.</p> <p>Each sequence is executed in order.</p>

The job could have been organized differently; the immediate segments could have been combined into one segment, and the same effect would have been achieved. The partitioning in the above example illustrates how a job can be organized and partitioned into related groups of job commands. This partitioning does not add any run-time overhead.

6.7 MARKING JOB CONTROL AND ADMINISTRATION

After a session has been created, job data can be sent to an SMC using the [sendStreamData \(overload 1\)](#) method or the [sendStreamData \(overload 2\)](#) method.

Note: Job data is created in XML format. A session is created using the [loginSession](#) method.

6.7.1 sendStreamData (overload 1)

Purpose	Sends streaming data to an SMC device session	
Syntax	uint sendStreamData(string pstrData uint uiTimeout)
Arguments	pstrData	The data sent to the SMC device. The string supplied contains an XML representation of the data.
	uiTimeout	Duration for attempting call in seconds. The special case of zero means to wait an infinite duration.

Comments	<p>Marking jobs are specified as sequences of data that represent instructions to the controller to:</p> <ul style="list-style-type: none"> set operational parameters activate the laser steering galvos in both marking and non-marking modes interact with external devices send event information back to a listening application <p>Job execution by the controller starts as soon as the job data is received by the module and continues for as long as job data is available. Very large jobs can be partitioned into logical chunks, such as at marking object boundaries, and streamed out to the device as buffering on the host and target allow. Since the execution of the job—and the process of streaming the data of the job—are asynchronous and overlapped, it is possible to maintain continuous job execution with no pauses.</p> <p>It is recommended that for very large streaming jobs, the job commands be packetized into groups of ~1000 instructions and each packet sent with a separate call to <code>sendStreamData()</code>. This minimizes the startup latency of executing the job and maximizes the use of the network and SMC buffering system.</p>
See also	<u>sendStreamData2</u>

If a syntax error is detected in the XML job data, an `OnData` event is generated to relate back to the application the nature of the error. See Section **Error! Reference source not found. Error! Reference source not found.**

6.7.2 [sendStreamData \(overload 2\)](#)

Purpose	Sends streaming data to an SMC device session	
Syntax	<pre>uint sendStreamData(string pstrData uint uiTimeout bool bWaitForACK out uint executionTime)</pre>	
Arguments	pstrData	The data sent to the SMC device. The string supplied contains an XML representation of the data.

	uiTimeout	Duration for attempting call in seconds. The special case of zero means to wait an infinite duration.
	bWaitForACK	If set to TRUE, the function does not return until a reception acknowledgement is received from the SMC. Otherwise, data packets are queued for execution.
	executionTime	Returns an estimated execution time in milliseconds for streaming style packets
Comments	<p>If a syntax error is detected in the XML job data, an OnData event is generated to relate back to the application the nature of the error.</p> <p>Marking jobs are specified as sequences of data that represent instructions to the controller to set operational parameters, to activate the laser steering galvos in both marking and non-marking modes, to interact with external devices, and to send event information back to a listening application. The job data is specified in an XML string, which is defined in the Streaming Job Data Definition section.</p> <p>Job execution by the controller starts as soon as the job data is received by the module and continues for as long as job data is available. Very large jobs can be partitioned into logical chunks, such as at marking object boundaries, and streamed out to the device as buffering on the host and target allow. Since the execution of the job and the process of streaming the data of the job are asynchronous and overlapped, it is possible to maintain continuous job execution with no pauses.</p> <p>It is recommended that for very large streaming jobs, the job commands be packetized into groups of ~1000 instructions and each packet sent with a separate call to sendStreamData(). This minimizes the startup latency of executing the job and maximizes the use of the network and SMC buffering system.</p>	
See also	<u>sendStreamData (overload 1)</u>	

If a syntax error is detected in the XML job data, an OnData event is generated to relate back to the application the nature of the error. See Section **Error! Reference source not found. Error! Reference source not found.**

6.7.3 `sendCorrectionData` (overload 1)

Purpose	Send a correction table to the board with transformations applied.	
Syntax	<div> uint sendCorrectionData(</div> <div> uint uiTableID string pstrCorrTablePath double dM00 double dM01 double dM10 double dM11 double dDx double dDy uint uiTimeout bool bWaitForACK) </div>	
Arguments	uiTableID	Table ID: 1, 2, 3, 4
	pstrCorrTablePath	Full path to Correction table data
	dM00	2x2 Matrix coefficient
	dM01	2x2 Matrix coefficient
	dM10	2x2 Matrix coefficient
	dM11	2x2 Matrix coefficient
	dDx	X offset (mm)
	dDy	X offset (mm)
	uiTimeout	Timeout for transaction
	bWaitForACK	Wait for ack from server

Comments	<p>Normally correction tables are automatically loaded for use when the SMC powers up. This method permits overriding the default tables with new ones which can be altered using the transform parameters.</p> <p>Both Cambridge Technology XML and Scanlab 2D CTB file formats are supported.</p> <p>Note that tables loaded using this method are not permanent and are lost after a SMC power-cycle.</p>
See also	sendCorrectionData (overload 2) , sendCorrectionData (overload 3)

6.7.4 [sendCorrectionData \(overload 2\)](#)

Purpose	Send a correction table to the board with transformations applied.	
Syntax	<pre>uint sendCorrectionData (uint uiTableID string pstrCorrTablePath double dScaleX double dScaleY double dRotation double dDx double dDy uint uiTimeout bool bWaitForACK)</pre>	
Arguments	uiTableID	Table ID: 1, 2, 3, 4
	pstrCorrTable Path	Full path to Correction table data
	dScaleX	X scale factor
	dScaleY	Y scale factor
	dRotation	Rotation in degrees (Positive is counter-clockwise)

	dDx	X offset (mm)
	dDy	X offset (mm)
	uiTimeout	Timeout for transaction
	bWaitForACK	Wait for ack from server
Comments	<p>Normally correction tables are automatically loaded for use when the SMC powers up. This method permits overriding the default tables with new ones which can be altered using the adjustment parameters.</p> <p>Both Cambridge Technology XML and Scanlab 2D CTB file formats are supported.</p> <p>Note that tables loaded using this method are not permanent and are lost after a SMC power-cycle.</p>	
See also	<u>sendCorrectionData (overload 1)</u> , <u>sendCorrectionData (overload 3)</u>	

6.7.5 [sendCorrectionData \(overload 3\)](#)

Purpose	Send a correction table to the board with transformations applied.	
Syntax	uint sendCorrectionData(uint uiTableID string pstrCorrTablePath double dScaleX double dScaleY double dRotationX double dRotationY double dRotationZ double dDx double dDy double dDz uint uiTimeout bool bWaitForACK)

Arguments	uiTableID	Table ID: 1, 2, 3, 4
	pstrCorrTablePath	Full path to Correction table data
	dScaleX	X scale factor
	dScaleY	Y scale factor
	dRotationX	Rotation in degrees about the X axis (Tip) (Positive is counter-clockwise)
	dRotationY	Rotation in degrees about the Y axis (Tilt) (Positive is counter-clockwise)
	dRotationZ	Rotation in degrees about the Z axis (Theta) (Positive is counter-clockwise)
	dDx	X offset (mm)
	dDy	Y offset (mm)
	dDz	X offset (mm)
	uiTimeout	Timeout for transaction
	bWaitForACK	Wait for ack from server
Comments	<p>Normally correction tables are automatically loaded for use when the SMC powers up. This method permits overriding the default tables with new ones which can be altered using the adjustment parameters.</p> <p>This method is designed to manipulate three-axis correction files.</p> <p>Note that tables loaded using this method are not permanent and are lost after a SMC power-cycle.</p>	
See also	sendCorrectionData (overload 1) , sendCorrectionData (overload 2)	

6.7.6 saveJobData

Purpose	Sends job data for storage in the SMC Flash memory or on an attached USB Flash storage drive	
Syntax	uint saveJobData(int iTargetLocation string pstrStorageName string pstrJobData uint puiAccessType uint puiTimeout)	
Arguments	iTargetLocation	Storage location: 0 = Local disk on the PC 1 = Flash on SMC 2 = USB Flash device on SMC 3 = tmp folder on SMC (volatile)
	pstrStorageName	Name to use as the file name. If iTargetLocation = 0, then this is an absolute path on the local machine.
	pstrJobData	XML representation of the job data
	puiAccessType	Access type: 0 = Overwrite 1 = Append (Reserved for future use)
	puiTimeout	Duration for attempting call in seconds

Comments	When a job has been constructed and tested using on-line workstation facilities, it can be sent to the SMC for storage on resident Flash memory or on an attached USB Flash storage drive. Jobs can also be stored on device tmp folder, which is volatile (job file will be lost after power cycle). Jobs stored on these devices can be run when the controller is placed in "local" mode.
See also	manageJobData , requestJobNameList , copyJobData

6.7.7 [sendJobData](#)

Purpose	Loads job data from local storage and sends it to the SMC for immediate execution	
Syntax	uint sendJobData(string pstrStorageName uint puiTimeout)
Arguments	pstrStorageName	Absolute path to the compiled job file on the local machine
	puiTimeout	Duration for attempting call in seconds
Comments	<p>Job data is loaded from a local drive and sent to the target SMC for immediate execution.</p> <p>When a job has been saved locally using the savejobData method, it can later be sent to the SMC for immediate execution</p>	
See also	manageJobData , requestJobNameList , saveJobData	

6.7.8 `copyJobData`

Purpose	Copies job data from local storage and sends it for storage in the SMC Flash memory or USB device	
Syntax	uint copyJobData(int iTargetLocation string pstrStorageName uint puiTimeout)	
Arguments	iTargetLocation	Storage location: 1 = Flash on SMC 2 = USB Flash device on SMC 3 = tmp folder on SMC (volatile)
	pstrStorageName	Absolute path to the compiled job file on the local machine
	puiTimeout	Duration for attempting call in seconds
Comments	When a job has been constructed and tested using on-line workstation facilities and saved locally using the saveJobData method, it can be sent to the SMC for storage on resident Flash memory or on attached USB Flash storage drives. Jobs stored on these devices can be run when the controller is placed in "local" mode.	
See also	manageJobData , requestJobNameList , saveJobData	

6.7.9 `manageJobData`

Purpose	Renames or deletes jobs that have been stored on the SMC	
Syntax	uint manageJobData(int iTargetLocation string pstrCurrentStorageName string pstrNewStorageName uint puiActionType uint puiTimeout)	

Arguments	iTargetLocation	Storage location: 1 = Flash on SMC 2 = USB Flash device on SMC 3 = tmp folder on SMC (volatile)
	pstrCurrentStorageName	Current file name
	pstrNewStorageName	New file name
	puiActionType	Action type: 0 = Delete 1 = Rename
	puiTimeout	Duration for attempting call in seconds
Comments	A job has been stored on the SMC can be renamed or deleted using this command.	
See also	saveJobData , requestJobNameList	

6.7.10 requestJobNameList

Purpose	Returns a list of jobs that have been stored on the SMC Flash or USB Flash	
Syntax	uint requestJobNameList(int iTargetLocation out int piJobCount out string pstrStorageName uint puiTimeout)	
Arguments	iTargetLocation	Storage location: 1 = Flash on SMC 2 = USB Flash device on SMC 3 = tmp folder on SMC (volatile)
	piJobCount	Number of jobs found on the target device

	pstrStorageName	File name of the data file. The file path is constructed by the API as follows: <PermStoragePath>\SMC\Config\<pstrStorageName>.xml where <u>PermStoragePath</u> is defined in the SysInfoData for the selected SMC and pstrStorageName is the name of the selected fixed data file as stored on the SMC without the ".xml" extension.
	puiTimeout	Duration for attempting call in seconds
Comments	Returns a list of jobs stored in the specified storage location on the SMC An example of the syntax of the list is as follows (for the SMC Flash device): <pre> <FlashJobList> <Job name='JobData.wlb' /> <Job name='LocalJob.wlb' /> </FlashJobList> </pre> If the device is specified to be the USB Flash device, then <FlashJobList> would be <USBJobList>.	
See also	<u>saveJobData</u> , <u>manageJobData</u>	

6.8 ASYNCHRONOUS COMMUNICATION

The SMC API uses programmed events to communicate asynchronous data back to an application. Events are divided into three types: Connect, Message and Data. Connect events are generated on major system state changes during login and logout operations. Message events are generated during normal execution of a job. They may be programmed to occur at specific points during job execution, or they may be generated by the system to signal an exception condition. Data events are created in response to specific application requests for data from the system, or from errors generated by the client API or SMC server firmware. This permits a non-blocking request/response code structure that is more efficient for data requests that take time to resolve.

6.8.1 OnConnectEvent

Purpose	Returns application and exception events from the SMC device session	
Syntax	OnConnectEvent(string pstrIPAddr, bool bState)
Arguments	pstrIPAddr	The IP address of the SMC whose connected state changed
	bState	True if connected; False if disconnected
Comments	The API can generate events when the API successfully "connects" to an SMC via the loginSession method or "disconnects" using the logoutSession method. These events are accessed via the OnConnectEvent command.	
See Also	loginSession , logoutSession	

6.8.2 OnMessageEvent

Purpose	Returns application and exception events from the SMC device session	
Syntax	OnMessageEvent(uint uiPayloadHigh, uint uiPayloadLow)
Arguments	uiPayloadHigh	Event type and data; encoded in two 16-bit entities: puiPayloadHigh[15..0] contains the event type described in Table 24 - OnMessageEvent Message Types puiPayloadHigh[31..16] contains event-type specific codes described in Table 25 - Predefined Application Message Event

	uiPayloadLow	Event data
Comments	<p>Jobs can use instructions that create "events" that can be sensed by an application. Events are also generated when exception conditions occur on the SMC.</p> <p>Events are used to communicate asynchronous data from the controller back to the application. Events are normally produced as a result of the controller executing a Begin Job, End Job, or Application Event instruction. Exception conditions may also create an event such as the response to an Abort message, servo error detection, etc. The data that classifies the event are passed back as two 32-bit payloads from the controller.</p>	
See Also	OnDataEvent	

Job messages are created using the [ApplicationEvent](#) job command. This command takes two arguments, the first of which is a user defined type code, and the second of which is an arbitrary 32-bit parameter. When this command is encountered by the marking engine controller, a Message Event is created. The message type code is passed back in [puiPayloadHigh\[31..16\]](#), and the parameter in [puiPayloadLow\[31..0\]](#). The system pre-defines some [ApplicationEvent](#) message type codes as indicated in *Table 24 - OnMessageEvent Message Types* on page 225.

Table 24 - ONMESSAGEEVENT MESSAGE TYPES

Message Type	Value	Description	puiPayloadHigh[31..16]
Reserved	All values not otherwise defined in this table	Reserved for future Cambridge Technology use	Reserved

Table 24 - ONMESSAGEEVENT MESSAGE TYPES

Message Type	Value	Description	puiPayloadHigh[31..16]
FixedDataProcessed	0x000F(15)	Fixed data update complete	0
JumpTimeCalDone	0x0013(19)	Calibration of jump-times complete	0
BeginJob	0x0041 (65)	The BeginJob instruction has been executed	0
EndJob	0x0042 (66)	The EndJob instruction has been executed	0
ApplicationEvent	0x5040 (20544)	User defined application event or predefined system application event (see Table 25 - Predefined Application Message Event)	User or predefined system specific

Application events are further refined by the uiPayloadHigh[31..16] value as defined in the following table.

Table 25 - PREDEFINED APPLICATION MESSAGE EVENT CODES

Application Event Code	Description	<u>uiPayloadLow[31..0]</u>
Reserved application event codes	Range 0x0000 – 0x0100 are reserved for CT use. All other codes not mentioned here are available to the user.	Varies

Table 25 - PREDEFINED APPLICATION MESSAGE EVENT CODES

Application Event Code	Description	<u>uiPayloadLow</u> [31..0]
0x0008 (8)	Digital Input Event. See SetDigitalInputConfig	Digital input bit-map of the pin that caused the event
0x0014 (20)	MOTF Trigger event	Value of the trigger counter when the pin state change was detected
Exception event codes	Generated by the SMC if exception conditions are detected at run-time. Code range between 0x2328 – 0x270F (9000 – 9999) are reserved for CT use.	Varies
0x2328 (9000)	Command processing was aborted	0
0x2329 (9001)	<u>Abort</u> message was processed	0
0x232A (9002)	Command FIFO empty time-out	0
0x232C (9004)	Bad opcode was received	0
0x232E (9006)	<u>WriteDigital</u> bad argument	0
0x232F (9007)	<u>LaserPower</u> bad argument	0
0x2330 (9008)	<set id='ActiveCorrectionTable'> bad argument	0
0x2331 (9009)	<set id='LaserPulse'> bad argument	0
0x2332 (9010)	<u>WaitForIO</u> bad argument	0
0x2333 (9011)	<u>WaitForIO</u> command time-out	0
0x2334 (9012)	<set id='LaserStandby'> bad argument	0
0x2336 (9014)	Time-out waiting for the laser to go active	0
0x2337 (9015)	<set id='MotfDirection'> bad argument	0
0x2338 (9016)	<MotfEnable> bad argument	0

Table 25 - PREDEFINED APPLICATION MESSAGE EVENT CODES

Application Event Code	Description	<u>uiPayloadLow</u> [31..0]
0x2339 (9017)	Performance Configuration File <u>Pulse Width</u> bad argument	0
0x233A (9018)	Performance Configuration File <u>Pulse</u> bad argument	0
0x233B (9019)	<set id='FieldOrientation'> bad argument	0
0x233D (9021)	Interlock was tripped	Interlock bit mask: <u>uiPayloadLow</u> [3..0] = Interlock[4..1]
0x233E (9022)	<u>WriteAnalog</u> bad argument	0
0x233F (9023)	<set id='TransformEnable'> bad argument	0
0x2342 (9026)	<set id='MotfMode'> bad argument	0
0x2343 (9027)	<u>RasterMode</u> not supported	0
0x2344 (9028)	<u>JobTimer</u> bad argument	0
0x2346 (9030)	An external Abort was processed	0
0x2393 (9107)	<set id='JumpAbsList'> bad argument <set id='MarkAbsList'> bad argument	0
0x2394 (9108)	Settle check timeout (<u>JumpAndFireList</u>)	Status register value being tested
0x2395 (9109)	Laser On Time is zero in <u>JumpAndFireList</u>	0
0x2397 (9111)	GSBus/XY2 Status fault detected	Status register value being tested
0x2398 (9112)	L2INST GEN Memory creation failed	0
0x2399 (9113)	L2INST Invalid vector args	0
0x239A (9114)	L2INST Invalid Circle args	0

Table 25 - PREDEFINED APPLICATION MESSAGE EVENT CODES

Application Event Code	Description	<u>uiPayloadLow[31..0]</u>
0x239C (9116)	L2INST Invalid Point args	0
0x239D (9117)	L2INST Invalid Spiral args	0
0x239E (9118)	L2INST Servo params creation failed	0
0x239F (9119)	L2INST Vect params creation failed	0
0x23A0 (9120)	L2INST Circle params creation failed	0
0x23A1 (9121)	L2INST Point params creation failed	0
0x23A2 (9122)	L2INST Spiral params creation failed	0
0x23A3 (9123)	L2INST Laser params creation failed	0
0x23A4 (9124)	L2INST Output Vectors failed	0
0x23A5 (9125)	L2INST Output Circles failed	0
0x23A6 (9126)	L2INST Output Points failed	0
0x23A7 (9127)	L2INST Output Spirals failed	0
0x23A9 (9129)	Jump-time calibration was not performed	0
0x23AA (9130)	Jump failed to settle in open-loop	Number of points

Special Notes on Interlocks and Handling Exceptions

Exceptions generally indicate that something bad has happened and that marking operations should be terminated as quickly as possible. This is especially important when high-power lasers are involved. The SMC provides for fast controlled shut-down of laser operations whenever an exception is detected by the hardware. Breaks in the interlock connectivity can be conditioned to shut down the laser and galvo motions and generate an exception event to the host application to notify it that the break occurred.

When a conditioned interlock trips, or any other hardware-detectable exception condition occurs, the marking engine controller immediately stops processing the vector stream, turns off the laser, and stops the galvo motion. It then disables the Interlock sensing function to avoid repeated notifications and sends an exception event message to the host application. If an exception occurs, the job cannot be restarted from where it left off.

The Interlock sensing function must be re-enabled after the fault condition is cleared. The following figure illustrates a sample protocol for handling an interlock break.

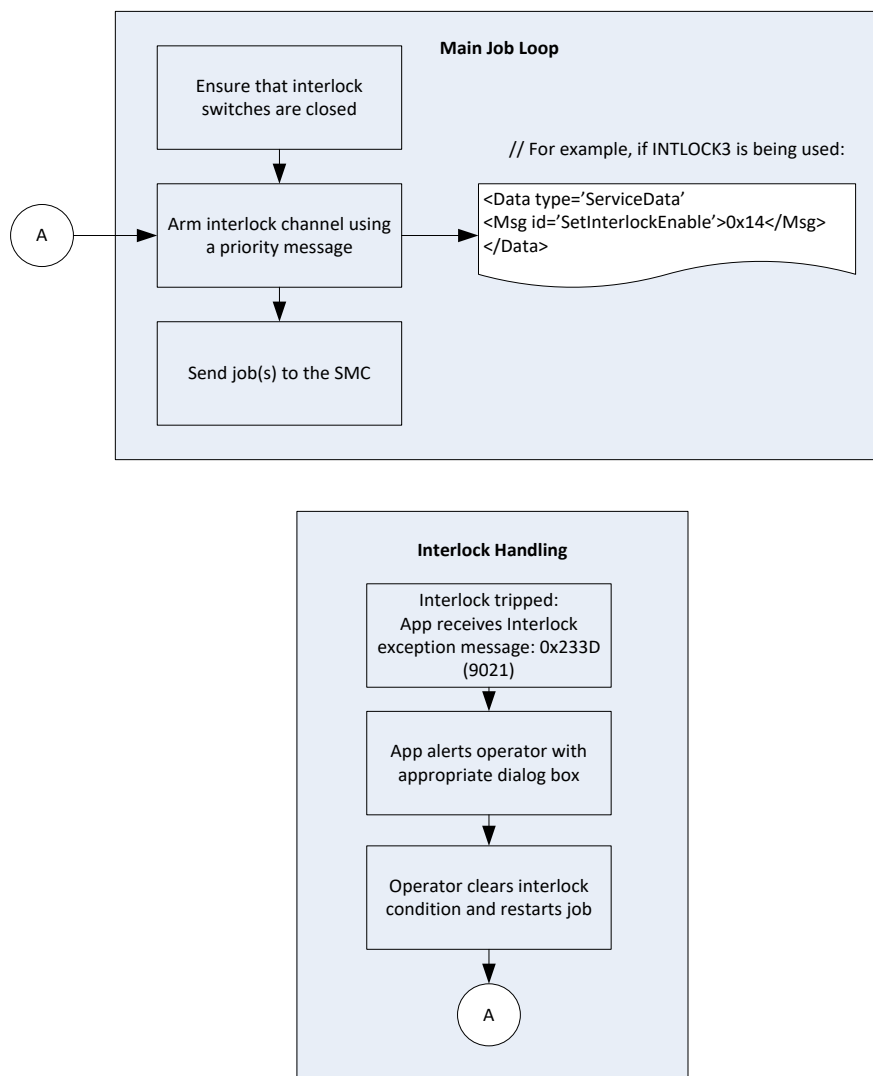


Figure 17 - INTERLOCK SEQUENCING

6.8.3 OnDataEvent

The OnDataEvent command is used to pass error details or requested data back to an application. Priority messages that return variable data do so by generating an OnData event. In general, a request for information is made by sending a Priority Data message (e.g., GetRegisters). When the SMC processes the message, it sends the requested data back through the OnData event channel.

The system will also generate a Data Event if there is a Job data syntax error. In this case, the suspected fragment of XML is returned as the event data along with an explanatory message.

OnDataEvent		
Purpose	Returns data requested from the SMC	
Syntax	OnDataEvent(uint uiDataID, uint uiErrorCode, string pstrData)
Arguments	uiDataID	Identifier of the data being returned. The identifiers are as follows; 0 - Reserved 1 - Client Errors 2 - Server Errors 3 - Registers Data 4 - Reserved
	uiErrorCode	Error code returned from the SMC; no error == 0
	pstrData	The data sent by the API. This data can originate from: <ul style="list-style-type: none"> •The API in the case of an XML command parsing error •The server in the case where a SW exception is detected •The SMC HW in the case where register data is requested The string supplied contains an XML representation of the data.

OnDataEvent	
Comments	<ul style="list-style-type: none"> • The system will generate a Data Event if there is a Job data syntax error. In this case, the suspected fragment of XML is returned as the event data along with an explanatory message. • See Error! Reference source not found. for an example of the type of data returned through this method.
See Also	<u>OnMessageEvent</u>

6.9 PRIORITY COMMUNICATION

Occasionally it may be necessary to send urgent commands to the controller that must bypass the data stream that is full of job data. sendPriorityData provides this mechanism. This mechanism is used to query an SMC for on-demand status information in cases where the cycle-time of broadcast packets is insufficient. It can also be used to pause/resume/abort a currently executing job.

6.9.1 sendPriorityData

sendPriorityData		
Purpose	Sends priority data to an SMC device session	
Syntax	uint sendPriorityData(string pstrData uint puiTimeout)
Arguments	pstrData	The data sent to the SMC device. The string supplied contains an XML representation of the priority request.
	puiTimeout	Duration for attempting call in seconds

sendPriorityData	
Comments	<p>An independent and parallel communication channel is provided to the controller to pass "out-of-band" commands. This channel of communication is used to send urgent commands to the controller, such as an <u>Abort</u> message or pause/resume messages.</p> <p>The method returns as soon as the message is sent, not when the operation is actually performed on the target. Some messages, however, create response events when the action is completed, such as "<u>Abort</u>" and "<u>GetRegisters</u>".</p>
See also	<u>getPriorityData</u>

6.9.2 PRIORITY MESSAGES

The following table contains descriptions of priority messages that can be sent using the sendPriorityData method.

Table 26 - PRIORITY MESSAGE DESCRIPTIONS

Message	Description/XML Example
Abort	<p>Abort based on the reason provided. This reason causes alternative action to be taken on the SMC device. Abort messages result in an On Message event being generated when the operation completes on the SMC. (Refer to Section Error! Reference source not found. ("Error! Reference source not found.") on page Error! Bookmark not defined. for more information.) The reason can be either of the following:</p> <ul style="list-style-type: none"> •Job - Abort the job that is currently running •Terminate - Abort the currently running job and terminate the current session connection <p>XML Example: <Data type='ServiceData' rev='1.1'> <Msg id='Abort' reason='Terminate'/> </Data></p>

Table 26 - PRIORITY MESSAGE DESCRIPTIONS

Message	Description/XML Example
Restart	<p>Performs a hardware reset of the SMC. The session will be disconnected and must be re-established before additional communications is possible.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='Restart'/> </Data></p>
Suspend	<p>Suspends the execution of the job. The job is paused at the next location where the lasers are off. If a <i>Mark</i> is currently in progress (including poly-vector mark), it is allowed to complete.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='Suspend'/> </Data></p>
Resume	<p>Job execution is permitted to continue.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='Resume'/> </Data></p>
GetRegisters	<p>Sends a request to the SMC to return the current values of several hardware registers on the module. Data is returned via a session OnData event message. (Refer to Section Error! Reference source not found. (“Error! Reference source not found.”) on page Error! Bookmark not defined. for more information.) The register data is parsed into named register entities if the attribute ‘raw’ is set to false (the default). If raw is set to true, the register data is returned in an indexed list of raw register values. See section Error! Reference source not found. (“GetRegisters Priority Message OnDataEvent Response”) on page Error! Bookmark not defined. for information regarding the returned data.</p> <p>Note: Raw lists are for advanced users only. (Please consult with Cambridge Technology Technical Support if you want to use raw lists.)</p> <p>XML Example: <Data type='ServiceData'> <Msg id='GetRegisters' raw='true'/> </Data></p>

Table 26 - PRIORITY MESSAGE DESCRIPTIONS

Message	Description/XML Example
SetInterlockEnable	<p><i>(Reserved for future use)</i> Enables or disables the interlock function of the SMC based on the "config" bit pattern.</p> <ul style="list-style-type: none"> • Bits[3..0] represent the interlock signals INTLOCK[4..1]. • A "1" enables a transition of the interlock signal going from the unasserted to the asserted state to generate an "Interlock" exception and shut down an active job provided that bit 4 is also asserted. • Bit[4] is the master enable bit for the interlock function. If this bit is set, then all enabled interlock signals should be de-asserted at power-up time or else an immediate "Interlock" exception will be generated when this parameter is processed. <p>If an interlock that is enabled is tripped, the condition that caused the trip must be cleared before a job can be restarted without generating another "Interlock" exception.</p> <p>The current state of the interlock physical signals can be seen in the Broadcast Status data as element <u>Interlock</u>.</p> <p>XML</p> <pre> <Data type='ServiceData'> <Msg id='SetInterlockEnable' config='0x14' /> </Data> or <Data type='ServiceData'> <Msg id='SetInterlockEnable'>0x14</Msg> </Data> </pre>

Table 26 - PRIORITY MESSAGE DESCRIPTIONS

Message	Description/XML Example
SetInterlockPolarity	<p><i>(Reserved for future use)</i> Sets the polarity of the interlock signals of the SMC based on the "config" bit pattern.</p> <p>Bits[3..0] represent the interlock signals INTLOCK[4..1]. A "1" corresponds to no current flowing through the interlock optical isolator. This condition is the interlock open state.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='SetInterlockPolarity' config='0x4'/> </Data> or <Data type='ServiceData'> <Msg id='SetInterlockPolarity'>0x4</Msg> </Data></p>
SetOffset	<p><i>(Obsolete)</i> Sets the run-time X, Y, and Z offsets to be applied to the vectors if the <u>TransformEnable</u> job command had been set to the enabled state.</p> <p>Otherwise, this message has no effect. The Z offset is optional and if not present it is not changed. Units are defined by the <set id='Units'> command</p> <p>XML Example: <Data type='ServiceData'> <Msg id='SetOffset'>200; 300; 100</Msg> </Data></p>
StopCurrentSequence	<p><i>(Reserved for future use)</i> Stops a continuously executing sequence at the end of its current iteration. Other sequences that are queued are run in order.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='StopCurrentSequence'></Msg> </Data></p>
StopAllSequences	<p><i>(Reserved for future use)</i> Stops a continuously executing sequence at the end of its current iteration. Any other sequences that are queued are not run.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='StopAllSequences'></Msg> </Data></p>

Table 26 - PRIORITY MESSAGE DESCRIPTIONS

Message	Description/XML Example
Flush	<p><i>(Reserved for future use)</i> Flushes all queued job data. Data that has reached the SMC marking engine is allowed to complete execution.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='Flush'></Msg> </Data></p>
SetRTJobTransform2D	<p>Sets the run-time coordinate transform {<i>Angle, Xoff, Yoff</i>} to be applied to the vectors if the <u>TransformEnable</u> job command has been set to the id value. If the <u>TransformEnable</u> job command has <i>not</i> been set to the id value, this message has no effect. The arguments are the following:</p> <p>id - 1 or 2 to select between two separate transform data sets</p> <p>Angle - Angle in degrees to rotate</p> <p>Xoff,Yoff - X and Y offset values to apply in user units</p> <p>XML <Data type='ServiceData'> <Msg id='SetRTJobTransform2D'>1; 25.0; 0.0; </Data></p>

Table 26 - PRIORITY MESSAGE DESCRIPTIONS

Message	Description/XML Example
SetRTJobTransformMatrix	<p>Sets the run-time coordinate transform matrix $\{M00, M01, M10, M11, Xoff, Yoff\}$ to be applied to the vectors if the <u>TransformEnable</u> job command had been set to the <i>id</i> value. If the <u>TransformEnable</u> job command has not been set to the <i>id</i> value, this message has no effect. The arguments are the following:</p> <p>id - 1 or 2 to select between two separate transform data sets</p> <p>M00 - - 2x2 transform matrix elements M11</p> <p>Xoff,Yoff - X and Y offset values to apply in user units. Offsets are applied after the matrix multiply operation.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='SetRTJobTransformMatrix'> 1; 0.707; -0.707; 0.707; 0.707; 5.0; 25.0 </Msg> </Data></p>
ExecuteJob	<p><i>(Reserved for future use)</i> Initiates the execution of a job previously stored on the SMC. The arguments are the following:</p> <p>location - 0 (local Flash), 1 (USB flash)</p> <p>mode - 0 (execute once), 1 ((execute continuous)</p> <p>name - File name of job stored on the SMC</p> <p>XML Example: <Data type='ServiceData'> <Msg id='ExecuteJob' location='0' mode='1' name='square.job'></Msg> </Data></p>

Table 26 - PRIORITY MESSAGE DESCRIPTIONS

Message	Description/XML Example												
SetDigitalInputConfig	<p>Sets the event-generating configuration of the digital inputs. The three arguments are encoded as follows:</p> <ul style="list-style-type: none">•arg1 - Bit-mask that enables a particular input to generate an event on a state change•arg2 and arg 3 - Polarity mask pair where the corresponding bit positions encode the event-generating behavior of the corresponding input as follows: <table><tr><th>arg2-bit</th><th>arg3-bit</th><th></th></tr><tr><td>0</td><td>0</td><td>Notify if transitioning to a low state</td></tr><tr><td>1</td><td>0</td><td>Notify if transitioning to a high state</td></tr><tr><td>X</td><td>1</td><td>Notify if transitioning to either state</td></tr></table> <p>The bit mappings to signals are as follows:</p> <p>bits[3..0] = AUX_GPI[4..1]_ISO</p> <p>bits[5..4] = AUX_START_ISO, START</p> <p>bits[9..6] = INTERLOCK[4..1] (LASER_STAT2, LASER_STAT1, LASER_STAT0, ABORT)</p> <p>bits[31..16] = EXTAUXIN[15..0]</p> <p>A transition from one to zero corresponds to a state of no current flowing through the isolator to a state of current flowing through the isolator.</p> <p>XML <Data type='ServiceData'> <Msg id='SetDigitalInputConfig'>0x0; 0x1; </Data></p>	arg2-bit	arg3-bit		0	0	Notify if transitioning to a low state	1	0	Notify if transitioning to a high state	X	1	Notify if transitioning to either state
arg2-bit	arg3-bit												
0	0	Notify if transitioning to a low state											
1	0	Notify if transitioning to a high state											
X	1	Notify if transitioning to either state											

Table 26 - PRIORITY MESSAGE DESCRIPTIONS

Message	Description/XML Example
GetCalFactors	<p>Retrieves the current calibration factors used by the SMC. See section Error! Reference source not found., “Error! Reference source not found.” for details on the information returned.Error! Reference source not found.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='GetCalFactors'!/> </Data></p>
SetCalFactors	<p>Set new calibration factors for use by the SMC. These override the X, Y, and Z calibration factors currently in use for active correction tables 1 & 2, respectively. These values are used to convert SMD, SMAPI, and ScanScript job coordinates to bits units used by the hardware. These values also update the API for when the <u>Units</u> command is not set to Bits units.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='SetCalFactors'> <XKFactor1>533.2332</XKFactor1> <YKFactor1>533.2332</YKFactor1> <ZKFactor1>650.1334</ZKFactor1> <XKFactor2>450.8765</XKFactor2> <YKFactor2>450.8765</YKFactor2> <ZKFactor2>520.4524</ZKFactor2> <XKFactor3>533.2332</XKFactor3> <YKFactor3>533.2332</YKFactor3> <ZKFactor3>650.1334</ZKFactor3> <XKFactor4>450.8765</XKFactor4> <YKFactor4>450.8765</YKFactor4> </Data></p>

Table 26 - PRIORITY MESSAGE DESCRIPTIONS

Message	Description/XML Example
SyncFileSystem	<p>Writes any file data buffered in memory on the SMC out to disk. After sending files to SMC using FTP or after other local (to the SMC) file operations, the content of the file may still be cached in memory and not actually be written to disk. If the power to the SMC is removed, the file content may be lost.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='SyncFileSystem' /> </Data></p>
StartLogging	<p>Send raw commands received by SMC marking engine to host computer for logging purpose. The host computer is identified by {hostIPAddress, port}.</p> <p> hostIPAddress : Host computer IPv4 address port: End port number</p> <p>XML Example: <Data type='ServiceData'> <Msg id='StartLogging' Port='5032' HostIPAddress='192.168.100.1' /> </Data></p>
StopLogging	<p>Stop sending raw commands received by SMC marking engine to host computer. It is paired with StartLogging priority message.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='StopLogging' /> </Data></p>
PowerScale	<p>Adjusts the laser power level by the value in the message. The laser power of an operating job will be immediately scaled by the factor upon receipt of the message.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='PowerScale'>0.9</Msg> </Data></p>

Table 26 - PRIORITY MESSAGE DESCRIPTIONS

Message	Description/XML Example
WriteDigital	<p>Writes the specified digital output port with the specified value. Port numbering and value descriptions are the same as for the job command <WriteDigital>.</p> <p>XML Example: <Data type='ServiceData'> <Msg id='WriteDigital'>0, 1</Msg> </Data></p>

6.9.3 getPriorityData

Some priority messages are designed to fetch information from the SMC on demand. Such information is returned to the application asynchronously through the use of Data Events (see Section **Error! Reference source not found.** (“**Error! Reference source not found.**”) on page **Error! Bookmark not defined.**). This asynchronous request/response scenario is not always convenient for an application, in which case the getPriorityData method can be used. The getPriorityData method directly returns the XML string representing the requested data without the application arming for a Data Event. The calling thread is blocked until the response packet has arrived from the board.

getPriorityData		
Purpose	Sends a priority data message to an SMC device session and waits for a response	
Syntax	uint getPriorityData(string pstrData out string pstrRegData uint puiTimeout)
Arguments	pstrData	A properly formed priority data request
	pstrRegData	The XML data returned by the SMC device
	puiTimeout	Duration for attempting call in seconds. Minimum 1 second.

getPriorityData	
Comments	<p>The message is sent and the requested data is returned to the application. This method blocks the calling thread until the data is returned or a timeout occurs.</p> <p>See Error! Reference source not found. for an example of the type of data returned through this method.</p>
See also	sendPriorityData

6.9.4 GetRegisters Priority Message OnDataEvent Response

<p>Data is returned asynchronous from the request.</p> <p>Register Data is returned as follows:</p> <p><Data type='HardwareState' rev='2.0'></p>	
<FpgaConfig>0xD</FpgaConfig>	// Advanced use only. Contact Cambridge Technology Technical Support.
<AuxIO_ID>0x0</AuxIO_ID>	// 0 == AuxDIO module not in use. 1 == module present.
<MOTFFrequency>0</MOTFFrequency>	// MOTF 0 speed (counts/10ms) // Deprecated. Use MOTF0Position
<MOTF0Frequency>0</MOTF0Frequency>	// MOTF 0 speed (counts/10ms)
<MOTF1Frequency>0</MOTF1Frequency>	// MOTF 1 speed (counts/10ms)
<ServoStatus>0x0</ServoStatus>	// Bits 6..3 == Z, Y, X Fault. Bits 2..0 == Z, Y, X Ready.
<XDAC>-500</XDAC>	
<YDAC>-500</YDAC>	
<ZDAC>0</ZDAC>	
<A1DAC>16</A1DAC>	
<A2DAC>0</A2DAC>	

<XY2Chan1>-500</XY2Chan1>	
<XY2Chan2>-500</XY2Chan2>	
<XY2Chan3>0</XY2Chan3>	
<XY2Status>0x0</XY2Status>	
<LaserTiming>50</LaserTiming>	
<LaserPower>0</LaserPower>	
<MOTFPosition>0</MOTFPosition>	// current MOTF 0 scaled count value // Deprecated. Use MOTF0Position
<MOTF0Position>0</MOTF0Position>	// current MOTF 0 scaled count value. If the <i>MotfCalFactor</i> is set to 1.0, this value is the actual encoder count for MOTF Port 0
<MOTF1Position>0</MOTF1Position>	// current MOTF 1 scaled count value. If the <i>MotfCalFactor</i> is set to 1.0, this value is the actual encoder count for MOTF Port 1
<DIO>0x3FF</DIO>	// bits[3..0] == AUX_GPI[4..1]_ISO bit[5..4] == AUX_START_ISO, START bits[9..6] == INTERLOCK[4..1] bits[13..10] == AUX_GPI4[4..1]_ISO bits[17..14] == JOBACTIVE, ERROR/NREADY, BUSY, LASING
<DIO.IN>0xF</DIO.IN>	// bits[3..0] == AUX_GPI[4..1]_ISO
<DIO.OUT>0x0</DIO.OUT>	// bits[3..0] == AUX_GPO[4..1]
<DIO.Control>0x1</DIO.Control>	// bits[4..0] == JOBACTIVE, ERROR/NREADY, BUSY, LASING, START
<DIO.Interlock>0xF</DIO.Interlock>	// bits[3..0] == INTLOCK[4..1]
<JobTimer>0</JobTimer>	
<XVectCmd>-500</XVectCmd>	
<YVectCmd>-500</YVectCmd>	
<ZVectCmd>0</ZVectCmd>	
<AuxIO_Ana1>0x20</AuxIO_Ana1>	// Optional auxiliary I/O module with analog sub-option

<AuxIO_Ana2>0x0</AuxIO_Ana2>	// Optional auxiliary I/O module with analog sub-option
<AuxIO_DIn>0x8FFC</AuxIO_DIn>	// Optional auxiliary I/O module
<AuxIO_DOut>0x0200</AuxIO_DOut>	// Optional auxiliary I/O module
</Data>	

6.9.5 GetCalFactors Priority Message OnDataEvent Response

<p>Data is returned asynchronous from the request.</p> <p>CalFactor Data is returned as follows:</p> <pre><Data type=CalFactors rev='1.0'> <XKFactor1>300.8149</XKFactor1> <YKFactor1>300.8149</YKFactor1> <ZKFactor1>231.518</ZKFactor1> <XKFactor2>300.8149</XKFactor2> <YKFactor2>300.8149</YKFactor2> <ZKFactor2>231.518</ZKFactor2> <XKFactor3>300.8149</XKFactor3> <YKFactor3>300.8149</YKFactor3> <ZKFactor3>231.518</ZKFactor3> <XKFactor4>300.8149</XKFactor4> <YKFactor4>300.8149</YKFactor4> <ZKFactor4>231.518</ZKFactor4> </Data></pre>	<p>These are the X, Y, and Z calibration factors currently in use for active correction tables 1 - 4, respectively. These values are used to convert SMD, SMAPI, and ScanScript job coordinates to bits units used by the hardware. These are also automatically read at session login time to initialize the API for when the <u>Units</u> command is not set to Bits units</p>
--	--

6.10 API ERROR CODES

Errors returned by the Session API are defined in the following table. The error descriptions can be accessed through the use of the method GetErrorCodeDescription.

GetErrorCodeDescription		
Purpose	Returns a string describing the meaning of the error code.	
Syntax	GetErrorCodeDescription (uint uiErrorCode)
Arguments	uiErrorCode	The error code returned by one of the API methods
Comments	Broadcast and Session methods return a code denoting the success (return value = 0), or failure of the method (return value != 0). A string describing the code can be fetched using this function.	
See Also		

7 REMOTE CONTROL API

There are three basic modes of operation for the SMC:

1. LAN-based streaming mode, where job data is managed on a host computer and sent to the SMC for immediate execution
2. Local mode, where an attached pendant is used to control the selection and execution of locally stored jobs
3. Remote mode, where a LAN-based supervisory interface can interact with the SMC and control all of the local mode functions

Remote mode is implemented as a text-based messaging interface over a normal TCP/IP socket connection. Messages are sent to the SMC as strings terminated with a line-feed character. All messages sent to the SMC are acknowledged with a line-feed terminated string.

All read or Get functions can be executed concurrently with other activities that the board may be performing, such as running jobs over the streaming interface. These functions would typically be associated with administrative functions such as examining passwords, networking parameters, job lists, etc. If modifications need to be made, or if actual execution control is required via the remote control interface, then a client application must "request control" or ownership of the module via the Remote Control API protocol command TakeHostControl.

7.1 TCP/IP INTERFACE

Remote control of the SMC can be established by any host computer that supports TCP/IP networking. This includes computers running Microsoft Windows, Linux, or other Unix derivatives. Communication with the board is established by opening a socket connection using the SMC IP address on port number 12500. The IP address can be learned by using the BroadcastAPIMethods to access the SysInfo data packets that are broadcast by the SMC. Alternatively, if the SMC is configured with a static IP address, broadcast monitoring is not required.

When a connection is established, the SMC transmits a "Welcome banner". This string must be read from the socket before bi-directional communication can be established.

7.2 RS232 INTERFACE

Remote control of the SMC can also be established by any host computer that supports RS232 serial communications. Communication is established by opening a COM port connection on the local computer that is connected to the SMC. The SMC COM port that is used for the protocol is controlled by settings in the Administration Configuration file. See the description of APIPort in *Table 10 - Administration Configuration* on page 45 for additional details.

If a single new-line character is sent to the remote control port, the SMC transmits a "Welcome banner". This string can be used to verify that communication has been established.

7.3 PROTOCOL SPECIFICATION

The following tables define the valid remote control commands and responses. Some commands take arguments. In such cases, the arguments are separated from the command and from each other by a "," (comma) character. If commands yield responses that have multiple values, the values are comma separated.

Note that all commands can be either text strings or numeric identifiers and are expressed in the table enclosed in quotes (" "). The quotation characters are NOT part of the command. This is also true for responses. Variable information is expressed as <variable> which is also a string.

There are two command modes can be used: basic mode and enhanced mode. All the commands listed in section 6.3.1 are expressed in basic mode.

The basic mode: the command string is sent to the SMC, and SMC sends back a response string. This mode poses a command-response synchronization problem when commands are send quickly. For example, the source sends command 1 to the SMC, and waits for response 1. If the SMC takes longer to respond, then the source may timeout and get no response. Next, the source may clear the receive buffer, send command 2 to SMC and waits for response 2. If at this time SMC completes command 1 and sends back response 1, then source will wrongly assume response 1 as response 2.

The enhanced mode: each command string is prefixed with a '\$' character, followed by a *UniqueNumber* chosen by the API user, followed by a ':' character. The SMC will use the same "\$*UniqueNumber*:" prefix with the response. For asynchronous event messages, the message is prefixed with "#*EventUniqueNumber*:" set of characters. *EventUniqueNumber* is generated

automatically by the SMC. If an event message has multiple lines, each line will have the same prefix. The enhanced command mode is the preferred mode. Below is a short example:

Command string	Response string	Note
\$1008:GetHostControlStatus	\$1008:5	Host not in control
\$1009:TakeHostControl	\$1009:0	Command success
...	...	
	#1002:65	Event BeginJob
	#1003:66	Event EndJob

Note also that all commands and arguments are case-sensitive.

RemoteAdministrator.exe is a sample program that uses the Remote API to access the SMC. It is located in C:\Program Files\Cambridge Technology\Client.

7.3.1 CONTROL AND COMMUNICATIONS COMMANDS

Abort (1) Command

Purpose	Stops the execution of a job
Implementation	"Abort" or "1"
Parameters	None
Returns	"0" – Command acknowledge
Comments	Immediately stops the execution of a running job and sets the JobRunning status to "Idle"
See also	N/A

TakeHostControl (2) Command

Purpose	Requests exclusive control of the SMC
---------	---------------------------------------

TakeHostControl (2) Command

Implementation	"TakeHostControl" or "2"
Parameters	None
Returns	"0" – Command acknowledge
Comments	This command will cause an abort of any actively running job. Use the GetJobStatus command to verify that the SMC job status is "Idle" before issuing this command.
See also	ReleaseHostControl , GetJobStatus

ReleaseHostControl (3) Command

Purpose	Releases exclusive control of the SMC to the LANStream host interface
Implementation	"ReleaseHostControl" or "3"
Parameters	None
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The TakeHostControl command is used to give the host exclusive control of the SMC.) • After this command is executed, jobs may be streamed to the SMC via the LANStream host interface.
See also	TakeHostControl

GetHostControlStatus (4) Command

Purpose	Returns the current SMC control status of this remote control session
Implementation	"GetHostControlStatus" or "4"
Parameters	None

GetHostControlStatus (4) Command

Returns	"125" – HOST_IN_CONTROL (control has been granted to this session) "126" – HOST_NOT_IN_CONTROL (this session is not in exclusive control of the SMC)
See also	TakeHostControl , ReleaseHostControl

GetHostInControl (5) Command

Purpose	Returns the current host interface that has exclusive control of the SMC
Implementation	"GetHostInControl" or "5"
Parameters	None
Returns	"Pendant" – Control has been granted to the pendant interface. "LANStream" – Control has been granted to the streaming LAN interface. "LAN" – Control has been granted to the LAN remote control interface.
See also	TakeHostControl , ReleaseHostControl

EnableBroadcasting (6) Command

Purpose	<i>(Obsolete)</i> Enables or disables the broadcast function of the SMC	
Implementation	"EnableBroadcasting <enable-state>" or "6, <enable-state>"	
Parameters	<enable-state>	Specifies whether the broadcast function of the SMC is enabled or disabled
	Value range	0 = Disabled 1 = Enabled
Returns	"0" – Command acknowledge	

EnableBroadcasting (6) Command

Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The TakeHostControl command is used to give the host exclusive control of the SMC.)
See also	TakeHostControl

LoadHardwareDefaults (7) Command

Purpose	Sets the current operating parameters of the SMC to their default values
Implementation	"LoadHardwareDefaults" or "7"
Parameters	None
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The TakeHostControl command is used to give the host exclusive control of the SMC.)
See also	TakeHostControl

HardwareReset (8) Command

Purpose	Forces a hardware reset of the SMC
Implementation	"HardwareReset" or "8"
Parameters	None
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The TakeHostControl command is used to give the host exclusive control of the SMC.) • The board will reboot as if power were just applied. Any IP addressing changes will be applied.

HardwareReset (8) Command

See also	TakeHostControl
----------	---------------------------------

GetRemoteIP (9) Command

Purpose	Returns the IP address of the LAN stream host that has control of the SMC
Implementation	"GetRemoteIP" or "9"
Parameters	None
Returns	Remote IP address in dot notation (e.g., 192.168.101.2)
Comments	If no host has control, the address "0.0.0.0" is returned.
See also	N/A

GetKFactor (10) Command

Purpose	Returns the calibration factor for the X-axis (in bits/mm) as stored in the correction table file assigned to CorrFile1 as defined in the Control Configuration file. Note: Unless a different Y-axis calibration factor is specified in the correction table file, the value returned by this command is also the Y-axis calibration factor.
Implementation	"GetKFactor" or "10"
Parameters	None
Returns	KFactor in floating-point notation
See also	GetYKFactor , GetZKFactor

SetPerformanceGlobals (14) Command		
Purpose	Sets factors to alter the run-time performance of the system.	
Implementation	"SetPerformanceGlobals	<mark-speed-adjust>,<laser-power-adjust>,<pulse-width-adjust>,<pulse-period-adjust>,<orientation>,<X-offset>,<Y-offset>,<Z-offset>" or
	"1 4	<mark-speed-adjust>,<laser-power-adjust>,<pulse-width-adjust>,<pulse-period-adjust>,<orientation>,<X- offset>,<Y-offset>,<Z-offset>"
Parameters	<mark-speed-adjust>	– Multiplier for <u>MarkSpeed</u> 0.5 - 1.5; specify "NOP" if no change is desired.
	<laser-power-adjust>	– Multiplier for <u>LaserPower</u> 0.8 - 1.2; specify "NOP" if no change is desired.
	<pulse-width-adjust>	– Multiplier for laser ON pulse width 0.5 - 1.5; specify "NOP" if no change is desired.
	<pulse-period-adjust>	– Multiplier for laser on pulse period 0.5 - 1.5; specify "NOP" if no change is desired.
	<orientation>	– Field orientation in degrees 0, 90, 180, 270; specify "NOP" if no change is desired.
	<X-offset>	– X-axis offset in bits -8388608 – 8388607; specify "NOP" if no change is desired. The offset can be specified in mm units by using a Decimal place "." in the number. Scaling to "bits" is done using head 1 calibration factors.
	<Y-offset>	– Y-axis offset in bits -8388608 – 8388607; specify "NOP" if no change is desired. The offset can be specified in mm units by using a Decimal place "." in the number. Scaling to "bits" is done using head 1 calibration factors.

SetPerformanceGlobals (14) Command		
	<Z-offset>	<p>Z-axis offset in bits -8388608 - 8388607; specify "NOP" if no change is desired.</p> <p>The offset can be specified in mm units by using a Decimal place "." in the number. Scaling to "bits" is done using head 1 calibration factors.</p>
Returns	"0" – Command acknowledge	
Comments	These factors alter the specified marking properties without the need for changing the job. These values are volatile and will not be valid if the SMC is reset.	
See also	ResetPerformanceGlobals	

ResetPerformanceGlobals (15) Command

Purpose	Resets the run-time performance modification parameters to their unity values.	
Implementation	"ResetPerformanceGlobals,<persist-to-file>" or "15,<persist-to-file>"	
Parameters	<persist-to-file>	Specifies whether to write reset values to the Performance Globals configuration file.
	Value range	<p>0 = Do not write reset values to the Performance Globals configuration file.</p> <p>1 = Write reset values to the Performance Globals configuration file.</p>
Returns	"0" – Command acknowledge	
Comments	The unity values result in no run-time modification to job-specified marking parameters.	
See also	SetPerformanceGlobals	

OpenCOMPort (16) Command		
Purpose	Opens the specified serial I/O COM port on the SMC	
Implementation	"OpenCOMPort,<port-ID>,<baud-rate>,<data-bits>,<parity>,<stop-bits>,<flow-control>" or "16,<port-ID>,<baud-rate>,<data-bits>,<parity>,<stop-bits>,<flow-control>"	
Parameters	<port-ID>	The serial I/O COM port to be opened
	Value range	0 = COM0 (OS console port on SMC main board) 1 = COM1 (Aux COM port on AUX-IO module) 2 = COM2 (Laser COM port) 3 = COM3 (N/A for SMC) 4 = COM4 (RS485 port on AUX-IO module)
	<baud-rate>	The baud rate for the serial I/O COM port specified in <port-ID>
	Value range	110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 128000, or 256000
	<data-bits>	The data bits for the serial I/O COM port specified in <port-ID>
	Value range	5, 6, 7, or 8
	<parity>	The parity for the serial I/O COM port specified in <port-ID>
	Value range	Even, Odd, None, Mark, or Space
	<stop-bits>	The stop bits for the serial I/O COM port specified in <port-ID>
	Value range	1, 1.5, or 2
	<flow-control>	The flow control for the serial I/O COM port specified in <port-ID>

OpenCOMPort (16) Command		
	Value range	None, XonXoff, CTS_RTS, or DSR_DTR
Returns	"0" – Command acknowledge	
Comments	<ul style="list-style-type: none"> • This command is available only if the system is configured for accepting streaming job data over Ethernet. The specified COM port is opened and is available for serial I/O. • This operation is intended to permit out-of-band communication to serial-port-based automation devices or laser systems. • A normal configuration might be specified as OpenCOMPort,2,38400,8,None,1,None • Only COM port-ID 1 has hardware flow control support. 	
See also	COMWriteLine , CloseCOMPort	

CloseCOMPort (17) Command		
Purpose	Closes a serial I/O COM port on the SMC	
Implementation	"CloseCOMPort,<port-ID>" or "17,<port-ID>"	
Parameters	<port-ID>	The serial I/O COM port to be closed
	Value range	0 = COM0 (OS console port on SMC main board) 1 = COM1 (Aux COM port on AUX-IO module) 2 = COM2 (Laser COM port) 3 = COM3 (N/A for SMC) 4 = COM4 (RS485 port on AUX-IO module)
Returns	"0" – Command acknowledge	
Comments	The COM port is closed and no longer available for serial I/O	
See also	COMWriteLine , OpenCOMPort	

COMWriteLine (18) Command		
Purpose	Writes the string argument to the COM port on the SMC.	
Implementation	"COMWriteLine,<port-ID>,<string>,<Timeout>" or "18,<port-ID>,<string>,<Timeout>"	
Parameters	<port-ID>	The serial I/O COM port to be written to
	Value range	0 = COM0 (OS console port on SMC main board) 1 = COM1 (Aux COM port on AUX-IO module) 2 = COM2 (Laser COM port) 3 = COM3 (N/A for SMC) 4 = COM4 (RS485 port on AUX-IO module)
	<string>	The string to be written to the COM port
	Value range	Any ASCII character string
	<Timeout>	Time to wait (in seconds) for a new-line terminated response
	Value range	0 - 65665
Returns	"<response string>" – Command acknowledge "ERROR_PORT_TIMEOUT" – The return string was not received before timeout expiration.	
Comments	This operation is intended to permit out-of-band communication to serial-port-based automation devices or laser systems. The specified port-ID must have been opened with the command <u>OpenCOMPort</u> .	
See also	<u>CloseCOMPort</u> , <u>OpenCOMPort</u>	

SetMotfEncoderRate (21) Command

Purpose	Sets the calibration factor used to convert encoder counts to laser galvo command bits (i.e., bits/encoder-count).
----------------	--

SetMotfEncoderRate (21) Command

Implementation	"SetMotfEncoderRate,<rate>"	
Parameters	<rate>	Bits per encoder-count
	Value range	-32768.0 to 32767.0
Returns	"0" – Command acknowledge	
Comments	The encoder rate relates encoder counts to how far an object travels in the lens field in galvo command bits. In the XML API, this is referred to as <u>MotfCalFactor</u>	
See also	N/A	

GetZKFactor (27) Command

Purpose	Returns the calibration factor for the Z-axis (in bits/mm) as stored in the correction table file assigned to CorrFile1 as defined in the Control Configuration file.
Implementation	"GetZKFactor" or "27"
Parameters	None
Returns	ZKFactor in floating-point notation
See also	<u>GetYKFactor</u> , <u>GetKFactor</u>

GetYKFactor (28) Command

Purpose	Returns the calibration factor for the Y-axis (in bits/mm) as stored in the correction table file assigned to CorrFile1 as defined in the Control Configuration file.
Implementation	"GetYKFactor" or "28"
Parameters	None

GetYKFactor (28) Command

Returns	YKFactor in floating-point notation
See also	GetKFactor , GetZKFactor

GetControllerTemp (29) Command

Purpose	Returns the temperature of the SMC board.
Implementation	"GetControllerTemp" or "29"
Parameters	None
Returns	The board temperature in degrees C.
See also	

COMReadLine (30) Command

Purpose	Reads a string from a COM port on the SMC.	
Implementation	"COMReadLine,<port-ID>,<Timeout>" or "30,<port-ID>,<Timeout>"	
Parameters	port-ID	Numeric port identifier
	Value range	2 == COM2 3 == COM3
	Timeout	How long to wait for a line terminator
	Value range	0 - 100
Returns	<Response string>	

COMReadLine (30) Command

Comments	This operation is intended to permit out-of-band communication to serial port based automation devices or laser systems. The specified port-ID must have been opened with the command OpenCOMPort. Lines are expected to be terminated with the new-line character.
See also	CloseCOMPort , OpenCOMPort

GetDigitalPort (35) Command

Purpose	Returns the state of the specified digital port	
Implementation	"GetDigitalPort, <PortID>" or "35, <PortID>"	
Parameters	port-ID	Numeric port identifier
	Value range	0 == Current Digital I/O port 1 == Auxiliary I/O port
Returns	<PortValue> (in hexadecimal notation, e.g. 0x1000233F)	
Comments	Current digital I/O port bits are decoded as described in section Error! Reference source not found. "Error! Reference source not found.", packet tag CurrentDIO. Auxiliary I/O bits are concatenated as 0x<AUX_GPO[15..0]><AUX_GPI[15..0]>	
See also		

GetCalScaleFactors (39) Command

Purpose	<i>(Reserved for future use)</i> Returns the current scale factors used to adjust the Cal Factor values that affect the job geometry size	
Implementation	"GetScaleFactors" or "39"	
Parameters	N/A	
Returns	"<XScale>, <YScale>, <ZScale>" (in floating-point notation)	

GetCalScaleFactors (39) Command

Comments	These factors are used to alter the values of KFactor, YKFactor, and ZKFactor when ScanMaster Designer, ScanMaster API, or ScanSript based jobs are run.
See also	SetScaleFactors

SetCalScaleFactors (40) Command

Purpose	<i>(Reserved for future use)</i> Sets the scale factors used to adjust the Cal Factor values that affect the job geometry size	
Implementation	"SetScaleFactors, <XScale>, <YScale>, <ZScale> " or "40, <XScale>, <YScale>, <ZScale>" (in floating-point notation)	
Parameters	XScale	Multiplier for KFactor
	Value range	0.0 – 10.0
	YScale	Multiplier for YKFactor
	Value range	0.0 – 10.0
	ZScale	Multiplier for ZKFactor
	Value range	0.0 – 10.0
Returns	"0" – Command acknowledge	
Comments	These factors are used to alter the values of KFactor, YKFactor, and ZKFactor when ScanMaster Designer, ScanMaster API, or ScanSript based jobs are run.	
See also	GetScaleFactors	

7.3.2 JOB EXECUTION CONTROL

ClearJobList (200) Command

Purpose	Clears the list of loaded jobs.
Implementation	"ClearJobList" or "200"
Parameters	None
Returns	"0" – Command acknowledge
Comments	Before jobs can be executed locally, they must be loaded from Flash memory into the SMC working memory. To change the list of loaded jobs, the list must be cleared first using this command. Jobs are loaded into the SMC Flash file system through the use of the saveJobData method.
See also	saveJobData

GetFlashJobFileList (203) Command

Purpose	Returns a comma-separated list of job files located on the SMC Flash drive.
Implementation	"GetFlashJobFileList" or "203"
Parameters	None
Returns	<job-list> – A comma-separated list of job names
Comments	Jobs are loaded into the SMC Flash file system through the use of the saveJobData method.
See also	saveJobData , and LoadFlashJob

GetUSBJobFileList (204) Command (Not yet supported)

Purpose	Returns a comma separated list of job files located on the USB Flash drive on the SMC.
Implementation	"GetUSBJobFileList" or "204"

GetUSBJobFileList (204) Command

(Not yet supported)

Parameters	None
Returns	<job-list> – A comma-separated list of job names
Comments	Jobs are loaded onto a USB Flash file system through the use of the saveJobData method.
See also	saveJobData

LoadFlashJob (205) Command

Purpose	Loads a job from the SMC Flash drive.
Implementation	"LoadFlashJob,<job-name>" or "205,<job-name>"
Parameters	<job-name> – The name of a job stored on the SMC
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The TakeHostControl command is used to give the host exclusive control of the SMC.) • The job name must include the extension as part of the name (e.g. "Circle.wlb").
See also	GetFlashJobFileList

LoadUSBJob (206) Command

(Not yet supported)

Purpose	Loads a job from the USB Flash drive on the SMC.
Implementation	"LoadUSBJob,<job-name>" or "206,<job-name>"
Parameters	<job-name> – The name of a job stored on the USB Flash file system device
Returns	"0" – Command acknowledge

LoadUSBJob (206) Command

(Not yet supported)

Comments	<ul style="list-style-type: none"> • The host must have exclusive control of the SMC (TakeHostControl) before issuing this command. • The job name must include the extension as part of the name (e.g. "Circle.wlb").
See also	GetUSBJobFileList

ExecuteJobOnce (207) Command

(Partially supported)

Purpose	Starts the one-time execution of a job.
Implementation	"ExecuteJobOnce, <job-name>" or "207, <job-name>"
Parameters	<job-name> – One of the jobs loaded with LoadFlashJob or LoadUSBJob . Note: Job-name is not currently supported. All of the loaded jobs are executed in the order they were loaded regardless if job-name is present or empty.
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • Before this command is executed, the host must have done the following: <ul style="list-style-type: none"> ○ Taken control of the SMC using the TakeHostControl command ○ Loaded a locally stored job with the LoadFlashJob command or the LoadUSBJob command • Unless the job was constructed with a WaitForIO instruction, it will begin to execute immediately. • The job can be stopped at any time by issuing an Abort command. • This command returns as soon as the job is dispatched.
See also	TakeHostControl , GetJobStatus , Abort , LoadFlashJob , LoadUSBJob

ExecuteJobContinuous (208) Command

(Partially supported)

Purpose	Starts the execution of a job and repeats it forever.
---------	---

ExecuteJobContinuous (208) Command

(Partially supported)

Implementation	"ExecuteJobContinuous, <job-name>" or "208, <job-name>"
Parameters	<job-name> – One of the jobs loaded with LoadFlashJob or LoadUSBJob Note: Job-name is not currently supported. All of the loaded jobs are executed in the order they were loaded regardless if job-name is present or empty.
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • Before this command is executed, the host must have done the following: <ul style="list-style-type: none"> ○ Taken control of the SMC using the TakeHostControl command ○ Loaded a locally stored job with the LoadFlashJob command or the LoadUSBJob command • The will begin to execute immediately. • If job execution is required to be synchronous with an external input such as STRTMRK, then it should have been constructed with a WaitForIO instruction after the BeginJob instruction. • At the completion of the job, the job will loop until an Abort command is received. • This command returns as soon as the job is dispatched.
See also	TakeHostControl , GetJobStatus , Abort , LoadFlashJob , LoadUSBJob

GetJobStatus (209) Command

Purpose	Returns the status of the currently executing job
Implementation	"GetJobStatus" or "209"
Parameters	None
Returns	"Idle" – No job is executing; a job may or may not be loaded. "Busy" – A job is executing.
See also	N/A

GetJobState (211) Command

Purpose	<i>(Reserved for future use)</i> Returns the state of the currently executing job
Implementation	"GetJobState" or "211"
Parameters	None
Returns	<p><current-sequence-index> – The index number of the currently executing sequence</p> <p><current-sequence-count> – Number of iterations of the current executing sequence</p> <p><current-segment-index> – The index number of the currently executing segment</p> <p><current-segment-count> – Number of iterations of the currently executing segment</p> <p><current-segment-name> – The name of the currently executing segment</p> <p>Ex: "1,2,3,1,Preamble"</p>
See also	N/A

GetJobElapsedTime (212) Command

Purpose	Returns the last measured duration (in milliseconds) of the currently executing job.
Implementation	"GetJobElapsedTime" or "212"
Parameters	None
Returns	<time-in-msec> – Last measured job execution duration in milliseconds
Comments	Time is measured based in the monitoring of the BeginJob and EndJob events. Jobs must be constructed with these instructions to be measured.
See also	N/A

StartScanScript (213) Command

Purpose	Indicates that a script body is to follow
Implementation	"StartScanScript" or "213"
Parameters	None
Returns	"0" – Command acknowledge
Comments	<p>This command is only available in enhanced command mode, and paired with <i>EndScanScript</i> command.</p> <p>Any text between <i>StartScanScript</i> and <i>EndScanScript</i> without a command prefix is considered as script body. Consult ScanMaster Designer (SMD) ScanScript help document for script syntax. A short example that makes 2 circles:</p> <pre>\$1001: StartScanScript Image.Circle(0, 0, 1) Image.Circle(-1, 0, 2) \$1002:EndScanScript</pre>
See also	EndScanScript , and RunScanScript

EndScanScript (214) Command

Purpose	Close the script body that started with StartScanScript command.
Implementation	" EndScanScript " or "214"
Parameters	None
Returns	"0" – Command acknowledge
Comments	<p>This command is only available in enhanced command mode, and paired with <i>StartScanscript</i> command.</p> <p>Any text between <i>StartScanScript</i> and <i>EndScanScript</i> without a command prefix is consider as script body. Consult ScanMaster Designer (SMD) ScanScript help document for script syntax. A short example that makes 2 circles:</p> <pre>\$1001: StartScanScript Image.Circle(0, 0, 1)</pre>

EndScanScript (214) Command

	Image.Circle(-1, 0, 2) \$1002:EndScanScript
See also	StartScanScript , and RunScanScript

RunScanScript (215) Command

Purpose	Run the script that loaded with StartScanScript and end EndScanScript command.
Implementation	" RunScanScript " or "215"
Parameters	None
Returns	"0" – Command acknowledge
Comments	This command is only available in enhanced command mode.
See also	StartScanScript , and EndScanScript

GetJobFileList (217) Command

Purpose	Returns a comma separated list of job files located on the tmp folder on the SMC.
Implementation	"GetJobFileList", <job-location> or "217, <job-location>"
Parameters	<job-location> – where to get the job list 1 = from SMC flash drive 2 = from SMC USB drive 3 = from SMC tmp folder
Returns	<job-list> – A comma-separated list of job names
Comments	Jobs are loaded onto a USB Flash file system through the use of the saveJobData method. tmp folder is volatile. All the files in tmp folder will be lost after power cycle.

GetJobFileList (217) Command

See also	saveJobData
----------	-----------------------------

GetJobFileList (217) Command

Purpose	Returns a comma separated list of job files located on the tmp folder on the SMC.
Implementation	"GetJobFileList", <job-location> or "217, <job-location>"
Parameters	<job-location> – where to get the job list 1 = from SMC flash drive 2 = from SMC USB drive 3 = from SMC tmp folder
Returns	<job-list> – A comma-separated list of job names
Comments	Jobs are loaded onto a USB Flash file system through the use of the saveJobData method. tmp folder is volatile. All the files in tmp folder will be lost after power cycle.
See also	saveJobData

7.3.3 SYSTEM ADMINISTRATION COMMANDS

SetAdminPIN (500) Command

Purpose	<i>(Obsolete)</i> Sets the Administrator PIN (password).	
Implementation	"SetAdminPIN,<admin-pin>" or "500,<admin-pin>"	
Parameters	<admin-pin>	New administrator PIN as a numeric string
	Value range	000000 - 999999
Returns	"0" – Command acknowledge	

SetAdminPIN (500) Command

Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The TakeHostControl command is used to give the host exclusive control of the SMC.) • The Administrator PIN is used with the Pendant interface to protect access to administrator functions.
See also	GetUserPIN , SetUserPIN

GetAdminPIN (501) Command

Purpose	<i>(Obsolete)</i> Gets the current Administrator PIN (password)
Implementation	"GetAdminPIN" or "501"
Parameters	None
Returns	<admin-pin> – Administrator PIN as a numeric string
Comments	The Administrator PIN is used with the Pendant interface to protect access to administrator functions
See also	SetAdminPIN , GetUserPIN , SetUserPIN

SetDHCPMode (502) Command

Purpose	Sets the DHCP addressing mode
Implementation	"SetDHCPMode,<mode>" or "502,<mode>"
Parameters	<mode> – "Static" or "Autodetect"
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The TakeHostControl command is used to give the host exclusive control of the SMC.) • Static IP addressing parameters are set using the SetLocalIP, SetLocalGateway, and SetSubnetMask commands. The board must be reset before these settings take effect. • Automatic IP addressing mode causes the SMC to request an IP address from a DHCP server when it boots up. If no server

SetDHCPMode (502) Command

	responds within a time-out period, the SMC automatically assigns itself an IP address in the range 169.254.xxx.yyy with a net-mask value of 255.255.0.0.
See also	SetLocalIP , SetLocalGateway , SetSubnetMask

GetDHCPMode (503) Command

Purpose	Gets the current DHCP addressing mode
Implementation	"GetDHCPMode" or "503"
Parameters	None
Returns	"Static" – Means that Static IP addressing is used "Autodetect" – Means that Automatic DHCP-based addressing is used
Comments	<ul style="list-style-type: none"> • Static IP addressing is set using the SetLocalIP, SetLocalGateway, SetSubnetMask and SetDHCPMode command. The board must be reset before these settings take effect. • Automatic IP addressing mode causes the SMC to request an IP address from a DHCP server when it boots up. If no server responds within a time-out period, the SMC automatically assigns itself an IP address in the range 169.254.xxx.yyy with a net-mask value of 255.255.0.0.
See also	SetLocalIP , SetLocalGateway , SetSubnetMask , SetDHCPMode

SetLocalGateway (504) Command

Purpose	Sets the gateway IP address used by the SMC if the SMC is in static IP addressing mode.
Implementation	"SetLocalGateway,<gateway-address>" or "504,<gateway-address>"
Parameters	Gateway Address in dot notation (e.g., 192.168.101.2)
Returns	"0" – Command acknowledge

SetLocalGateway (504) Command

Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The TakeHostControl command is used to give the host exclusive control of the SMC.) • Other static IP addressing parameters are set using the SetLocalIP and SetSubnetMask commands. The board must be reset before these settings take effect.
See also	GetLocalGateway , SetLocalIP , SetSubnetMask , SetDHCPMode

GetLocalGateway (505) Command

Purpose	Returns the gateway IP address used by the SMC if the SMC is in static IP addressing mode.
Implementation	"GetLocalGateway" or "505"
Parameters	None
Returns	Gateway Address in dot notation (e.g., 192.168.101.2)
See also	SetLocalGateway

SetLocalIP (506) Command

Purpose	Sets the IP address used by the SMC if the SMC is in static IP addressing mode.
Implementation	"SetLocalIP,<IP-address>" or "506,<IP-address>"
Parameters	IP Address in dot notation (e.g., 192.168.101.200)
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The TakeHostControl command is used to give the host exclusive control of the SMC.) • Other static IP addressing parameters are set using the SetLocalGateway and SetSubnetMask commands. The board must be reset before these settings take effect.

SetLocalIP (506) Command

See also	GetLocalIP , SetLocalGateway , SetSubnetMask , SetDHCPMode
----------	--

GetLocalIP (507) Command

Purpose	Returns the IP address used by the SMC if the SMC is in static IP addressing mode.
Implementation	"GetLocalIP" or "507"
Parameters	None
Returns	Static IP Address in dot notation (e.g., 192.168.101.2)
See also	SetLocalIP

SetNodeFriendlyName (508) Command

Purpose	Sets the "friendly name" of the SMC.
Implementation	"SetNodeFriendlyName,<friendly-name>" or "508,<friendly-name>"
Parameters	<friendly-name> – String representing the friendly name assigned to the SMC
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The TakeHostControl command is used to give the host exclusive control of the SMC.) • The "friendly name" of the SMC corresponds to the tag FriendlyName in the Administration Configuration file.
See also	GetNodeFriendlyName

GetNodeFriendlyName (509) Command

Purpose	Returns the "friendly name" of the SMC.
Implementation	"GetNodeFriendlyName" or "509"

GetNodeFriendlyName (509) Command

Parameters	None
Returns	Friendly name – String representing the friendly name assigned to the SMC
Comments	The "friendly name" of the SMC corresponds to the tag <u>FriendlyName</u> in the Administration Configuration file.
See also	<u>SetNodeFriendlyName</u>

SetSubnetMask (510) Command

Purpose	Sets the subnet mask used by the SMC if the SMC is in static IP addressing mode.
Implementation	"SetSubnetMask,<mask>" or "510,<mask>"
Parameters	<mask> – Subnet mask in dot notation (e.g., 255.255.255.0)
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The <u>TakeHostControl</u> command is used to give the host exclusive control of the SMC.) • Other static IP addressing parameters are set using the <u>SetLocalGateway</u> and <u>SetLocalIP</u> commands. The board must be reset before these settings take effect.
See also	<u>GetSubnetMask</u> , <u>SetLocalGateway</u> , <u>SetLocalIP</u> , <u>SetDHCPMode</u>

GetSubnetMask (511) Command

Purpose	Returns the subnet mask used by the SMC if the SMC is in static IP addressing mode.
Implementation	"GetSubnetMask" or "511"
Parameters	None
Returns	Subnet mask in dot notation (e.g., 255.255.255.0)
See also	<u>SetSubnetMask</u>

SetUserPIN (512) Command

Purpose	<i>(Obsolete)</i> Sets the Administrator PIN (password)
Implementation	"SetUserPIN,<user-pin>" or "512,<user-pin>"
Parameters	<user-pin> – New user PIN as a numeric string
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • Before this command can be executed, the host must have exclusive control of the SMC. (The TakeHostControl command is used to give the host exclusive control of the SMC.) • The User PIN is used with the Pendant interface to protect access to SMC functions.
See also	GetAdminPIN , SetAdminPIN

GetUserPIN (513) Command

Purpose	<i>(Obsolete)</i> Gets the current User PIN (password)
Implementation	"GetUserPIN" or "513"
Parameters	None
Returns	<user-pin> (User PIN as a numeric string)
Comments	The User PIN is used with the Pendant interface to protect unauthorized access to SMC functions.
See also	SetUserPIN , GetAdminPIN , SetAdminPIN

SetCOMPortSpeed (514) Command

Purpose	Sets the speed of the pendant, API, and motion-control COM ports on the SMC.
---------	--

SetCOMPortSpeed (514) Command

Implementation	"SetCOMPortSpeed,<pendant-port-baud-rate>,<api-port-baud-rate>,<motion-control-port-baud-rate>" or "514,<pendant-port-baud-rate>,<api-port-baud-rate>,<motion-control-port-baud-rate>"	
Parameters	<pendant-port-baud-rate>	The speed of the pendant COM port on the SMC
	Value range	110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 128000, and 256000
	<api-port-baud-rate>	The speed of the API COM port on the SMC
	Value range	110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 128000, and 256000
	<motion-control-port-baud-rate>	The speed of the motion-control COM port on the SMC
	Value range	110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 128000, and 256000
Returns	"0" – Command acknowledge	
Comments	The three COM ports on the SMC are logically identified as “pendant”, “api”, and “motion-control” and are physically mapped using the command SetCOMPortAssignments .	
See also	GetCOMPortSpeed , SetCOMPortAssignments , GetCOMPortAssignments	

GetCOMPortSpeed (515) Command

Purpose	Gets the current speed of the pendant, API, and motion-control COM ports on the SMC.
Implementation	"GetCOMPortSpeed" or "515"

GetCOMPortSpeed (515) Command

Parameters	None
Returns	<p><pendant-port-baud-rate>,<api-port-baud-rate>,<motion-control-port-baud-rate></p> <p>Note: The possible values of <pendant-port-baud-rate>, <api-port-baud-rate>, and <motion-control-port-baud-rate> are the following: 110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 128000, and 256000.</p>
Comments	Each returned value can be any one of the following: 110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 128000, and 256000
See also	SetCOMPortSpeed , SetCOMPortAssignments , GetCOMPortAssignments

SetCOMPortAssignments (516) Command

Purpose	Maps the SMC COM ports to the logical pendant, api and motion-control ports
Implementation	"SetCOMPortAssignments,<pendant-port>,<api-port>,<motion-control-port>" or "516,<pendant-port>,<api-port>,<motion-control-port>"
Parameters	<p><pendant-port> – 0 or 1 <i>Reserved for future use</i></p> <p><api-port> – 0 or 1</p> <p><motion-control-port> – 1 or 4</p>
Returns	"0" – Command acknowledge
Comments	<ul style="list-style-type: none"> • The COM port assignments must be unique. • If hardware flow control is required, then COM1 (1) should be used. • This command updates the contents of the Administration Configuration file.
See also	SetCOMPortSpeed , GetCOMPortSpeed , GetCOMPortAssignments

GetCOMPortAssignments (517) Command

Purpose	Gets the current mapping of the SMC COM ports to the logical pendant, api and motion-control ports
Implementation	"GetCOMPortAssignments>" or "517"
Parameters	None
Returns	"<pendant-port>,<api-port>,<motion-control-port>" (1, 2, or 3)
See also	SetCOMPortSpeed , GetCOMPortSpeed , SetCOMPortAssignments

7.4 REMOTE CONTROL RETURN CODES

In certain cases, the response message may be an error message rather than the expected "0" (ACK) or return variable(s). Table 38 - Remote Control Return Codes in page 242 describes the codes that may be returned.

8 APPENDIX A - THEORY OF OPERATION

8.1 SCANNING JOB FUNDAMENTALS

The purpose of scanning jobs is to direct the motion of laser galvanometers while simultaneously modulating a laser beam. The laser is turned on when a pattern is to be drawn, and it is turned off when moving to the beginning of a new pattern location. In laser marker systems, the drawing action is commonly referred to as a “Mark”, and a move to new pattern location is called a “Jump”. These terms will be used in this appendix even though an SMC could be used for laser projection where a more appropriate term for “Mark” might be “display”.

8.1.1 COORDINATE SYSTEM CONVENTIONS

The movement commands “MarkAbs” and “JumpAbs” are expressed in Cartesian coordinates as shown in the following figure.

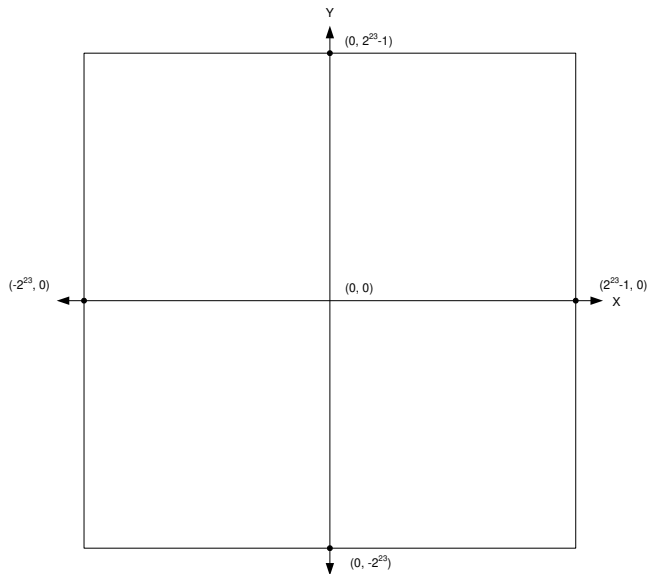


Figure 18 - SCANNING SYSTEM COORDINATE CONVENTIONS

The SMC is inherently a 24-bit address controller. Some commands however are backwards compatible to the 16-bit EC1000. If the API is operating in “bits” mode (see the [Units](#) command) then the arguments of the commands JumpAbs and MarkAbs are assumed to be 16-bit integers with a range of -32768 to +32767 and are converted to 24-bit values by padding them with 8 bits of zeros in the least significant bit positions.

If the motion commands JumpAbsEx and MarkAbsEx are used, the API must be guided how to make the conversion the 24-bits. This is because these commands can pass up to 32-bits of global coordinate data which could represent values in a 16-bit, 20-bit, or 24-bit scanner address space depending on the origins of the job data and what the design assumptions were. By using the command [ActuatorUnits](#), the API can correctly convert the command values to 24-bit form. By default, 16-bit data is assumed.

8.1.2 MARKS AND JUMPS

Laser marking is specified by a list of XML data that defines “Jumps” to locations and “Marks” to the end points of a vector or series of “connected” vectors otherwise known as poly-vectors. Other XML data represent commands to specify related actions and pauses required to ensure the desired marking quality. The terms Mark, Jump, and related delays are defined in *Table 27 - Laser Marking Terms and Definitions* on the following page.

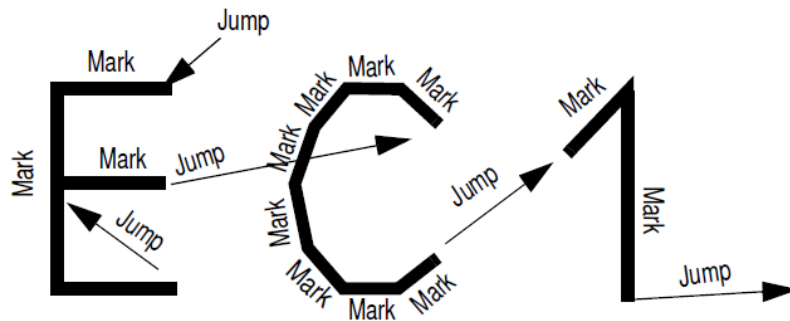


Figure 19 - LASER MARKING SAMPLE

The above figure shows a sample of the beginning of a simple laser marking. The image is composed of straight line segments (vectors). Connected line segments are formed with sequential Mark commands and spaces between unconnected segments are formed with Jump commands. Both Marks and Jumps are controlled-velocity coordinated X and Y galvo motions. The speeds are controllable within a job.

8.1.3 LASER MARKING TERMS AND DEFINITIONS

The following table contains definitions of laser marking terms.

Table 27 - LASER MARKING TERMS AND DEFINITIONS

Term	Purpose
Jump	<p>A jump causes a (typically) rapid movement of the scanner mirrors to a new position. Ideally, no marking occurs during a jump. The laser is typically turned off during a jump.</p> <p>If a jump is followed by a mark, the jump command defines the starting point (X and Y coordinates) of the laser marking; the SMC directs the laser to the end of the “jump” position where marking will begin.</p>
Jump Speed	The jump speed determines the speed of the jump. The laser is off during a jump, and the jump speed is set high enough to maximize throughput but low enough to minimize instability in the galvo motion as the galvo slows down in its approach to the next marking location.
Mark	A mark command begins the marking process. The laser typically turns on at the beginning of the mark command and continues at a set speed to its pre-defined location at the end point of the command. As show in the above figure, subsequent mark commands can create a sequence of marks. The laser is turned off at the end of the last mark command in a series of commands.
Mark Speed	The Mark Speed sets the speed during marking. The speed is set to a value that allows the laser to form the proper width and depth of a mark in the target media. This value is dependent on the laser power and target material.
Delays	Delays are used to ensure that the marking is complete with no skips, no over-burns, and no inadvertent marks. Delay commands are necessary to compensate for system inertia, acceleration, deceleration, and requested jump and marking speeds.

In addition to the dynamic signals used to control the galvanometers and lasers, the SMC provides supplemental digital inputs and outputs for external equipment synchronization, and two analog outputs for laser power adjustment. These signals can be manipulated at any point in a job, but are less tightly controlled in time than the galvanometer and laser control signals.

The default initial galvanometer position after system power-up is in the center of the image field unless otherwise specified in the ControlConfig file by the InitPosition tag. Marks and jumps are

specified from the current position of the galvanometers to a new target position. Jobs typically begin with an absolute jump to the first marking position, and after that, each vector (jump or mark) starts at the new current position, which is usually the end point of the preceding vector.

8.1.4 MICRO-VECTORIZING

Controlled velocity marking and jumping is accomplished through a process call micro-vectoring. This process is illustrated in the following figure. The marking engine of the SMC takes a vector and divides it into multiple shorter segments that are applied to the galvos at regularly spaced time intervals. This interval is known as the update interval. The galvo speed is controlled by the magnitude of the *change* in the output command at each update period.

The figure shows the sequence of typical output commands for the X-axis. The commands for the Y-axis and Z-axis are similar and are strictly locked in time with the X-axis, differing only in magnitude of the discrete steps. As the X-axis reaches successive targets X_1, X_2 , etc., so do the Y- and Z-axis reach their corresponding targets, Y_1, Z_1, Y_2, Z_2 , etc.

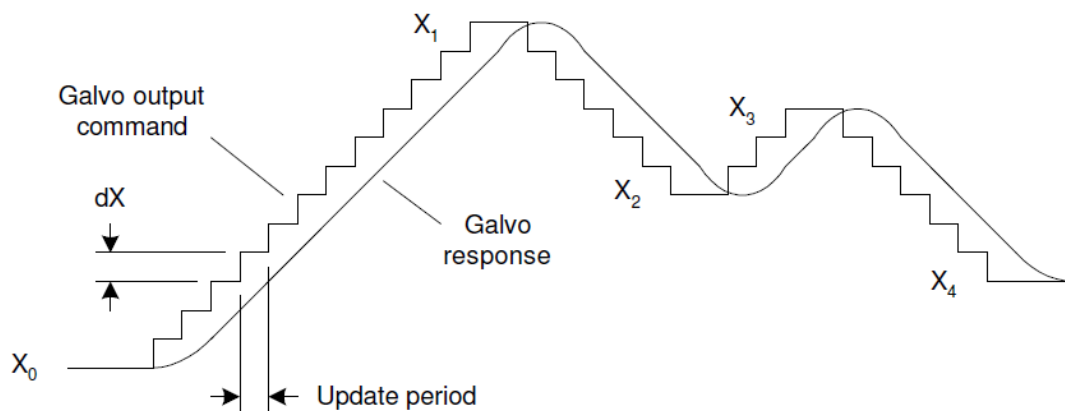


Figure 20 - MICRO-VECTOR OPERATION

8.1.5 DELAYS

Because laser scanning systems are electro-mechanical in nature, various delays must be employed to compensate for inertial effects of the mirror and motor structure. These inertial effects generally result in a positional lag of the deflection mirrors relative to the electrical command to make them move. These delays are used to properly time laser on/off and modulation signals relative to the

mirror positions. In addition to compensating for lag times, the delays can be used to compensate for transient instability in mirror positions after a step to a new location. The following figures illustrate these effects.

Each system configuration requires fine-tuning of delay commands to ensure full and complete marking with no overburns. The individual delay settings are dependant on the dynamic response of the galvo/mirror combination in use, and the sensitivity characteristics of the marking medium. Determining these delays is typically a process of trial and error. The delays are specified as part of the job definition described in the following pages.

Table 28 - DELAY PARAMETERS

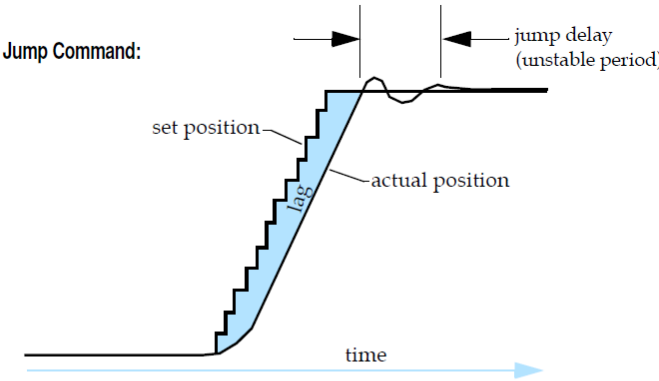
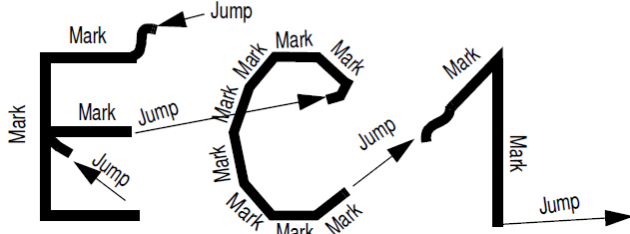
Parameter	Purpose	Effects
<u>JumpDelay</u>	<p>During a jump, the system mirrors accelerate to rapidly get to the next mark position—ideally at the fastest possible speed—to minimize overall marking time. As with all accelerations, mirror and system inertia create a slight lag at the beginning of the acceleration. Likewise, the system will require a certain delay (settling time) at the end of the jump as it decelerates to precisely the correct speed required for accurate marking.</p> <p>Acceleration and deceleration times and settling times will vary from system to system—due to the weight of mirrors, the type of galvanometer, etc.—and will also vary</p>	 <p>Jump Command:</p> <p>set position</p> <p>actual position</p> <p>lag</p> <p>time</p> <p>jump delay (unstable period)</p> <p>Jump Delay Too Short: Marking starts before mirrors properly settle</p> 

Table 28 - DELAY PARAMETERS

Parameter	Purpose	Effects
	<p>depending on the jump speed and the length of the jump.</p> <p>A too-short Jump Delay will cause marking to start before mirrors are properly settled, resulting in inadvertent marking.</p> <p>A too-long Jump Delay will have no visible effect, but marking is delayed so overall job production time (marking time) increases.</p>	
<u>MarkDelay</u>	<p>A mark delay at the end of marking a line segment allows the mirrors to move to the required position prior to executing the next mark command.</p> <p>A too-short Mark Delay will allow the subsequent jump command to begin before the system mirrors get to their final marking position. The end of the current mark will turn upwards towards the direction of the jump vector, as shown to the right.</p> <p>A too-long Mark delay will cause no visible</p>	<p>Mark Delay Too Short: Marking continues into a jump vector</p>

Table 28 - DELAY PARAMETERS

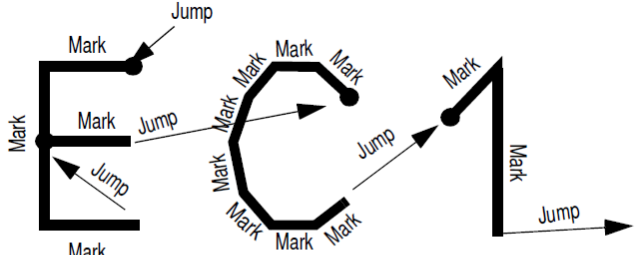
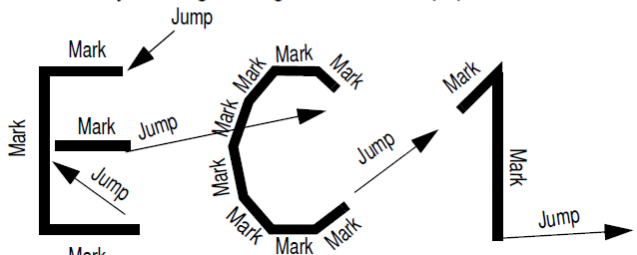
Parameter	Purpose	Effects
	marking errors but will add to the overall processing time.	
<u>LaserOnDelay</u>	<p>The Laser On Delay can be used to prevent burn-in effects at the start of a vector. This delay is typically used to turn on the laser after the first few microsteps of a mark command, ensuring that the laser's motion control systems (mirrors, etc.) are “up to speed” before marking. The vectors must be scanned with a constant velocity to ensure uniform marking.</p> <p>This delay can have either a positive or negative value and will vary with different marking media (some media require a burn-in time to begin marking). The goal is to adjust the Laser On Delay to ensure uniform marking with no variations of intensity throughout the desired vector.</p> <p>Typically, too short of a delay will cause burn-in effects, and too long of a delay may cause</p>	<p>Laser On Delay Too Short: burn-in at start points</p>  <p>The diagram illustrates three marking paths: a square, a circle, and a triangle. Each path starts with a 'Jump' arrow pointing to the first 'Mark' point. The paths are composed of multiple 'Mark' segments connected by 'Jump' arrows. The paths are shown with varying degrees of burn-in at the start points, indicating that the laser was on too long before the marking began.</p> <p>Laser On Delay Too Long: marking starts too late, skips points</p>  <p>The diagram illustrates the same three marking paths as above. In this case, the 'Jump' arrows are longer, indicating a longer delay before marking begins. This results in the laser skipping the initial part of the vector, leading to incomplete or skipped points at the start of each path.</p>

Table 28 - DELAY PARAMETERS

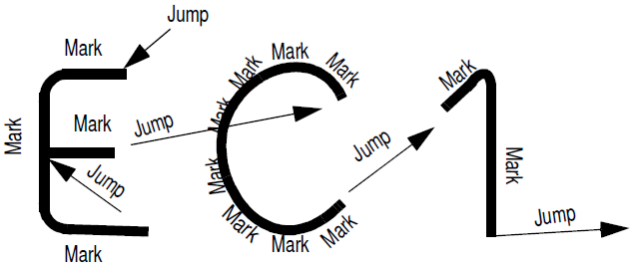
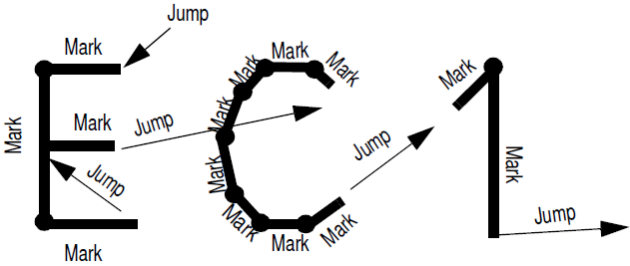
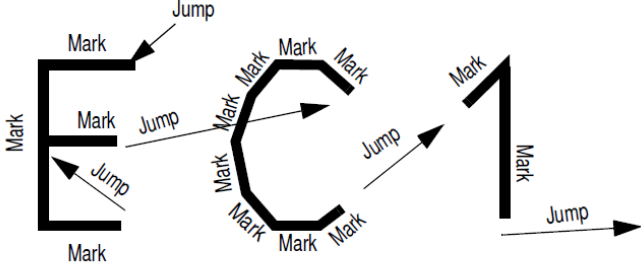
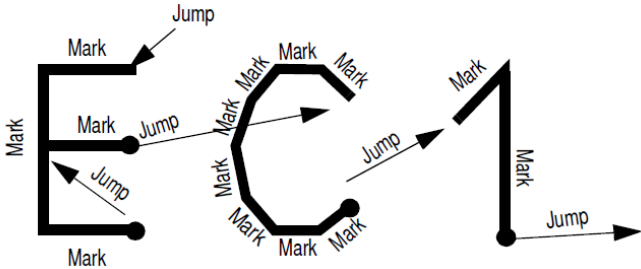
Parameter	Purpose	Effects
	disconnected line segments.	
<u>PolyDelay</u>	<p>A polygon delay is a delay that is automatically inserted between two marking segments. The minimum delay allows enough time for the galvos and mirror to “catch-up” with the command signal before a new command is issued to move on to the next point.</p> <p>If variable polygon delay mode is selected, then the delay is variable and changes as a function how large an angular change is required to move on to the next point. The larger the angular change, the longer it takes for the galvos to change direction and accelerate to the required speed in the new direction. The delay is scaled proportionally to the size of the angle.</p>	<p>Polygon Delay Too Short: characters not wellformed</p>  <p>Polygon Delay Too Long: burn-in at junctions in the vector</p> 

Table 28 - DELAY PARAMETERS

Parameter	Purpose	Effects
<u>LaserOffDelay</u>	<p>The Laser Off Delay can be used to prevent burn-in effects at the end of a vector. This delay is typically used to turn off the laser just after the last few micro-steps of a mark command, ensuring that the marking stops exactly where it is supposed to.</p> <p>The goal is to adjust the Laser Off Delay to ensure uniform marking with no variations of intensity throughout the desired vector.</p> <p>Typically, too short of a delay will cause line segments that are prematurely terminated, and too long of a delay will cause burn-in at the end of line segments.</p>	<p>Laser Off Delay Too Short: marking stops too soon, skipped endpoints</p>  <p>Laser Off Delay Too Long: marking stops too late, burn-in at end points</p> 

The relationship of the delays to the micro-vectoring process is illustrated in the following figure.

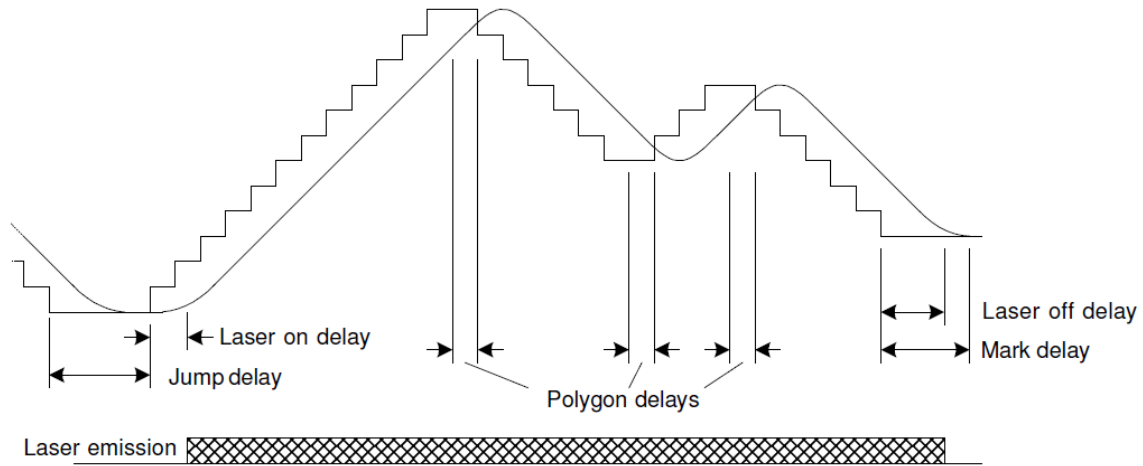


Figure 21 - MICRO-VECTURING AND LASER TIMING RELATIONSHIPS

8.2 IMAGE FIELD CORRECTION

Image field correction capability is provided to compensate for optical errors induced by all two-mirror laser beam systems. These optical distortions are caused by a number of factors, including the distance between each mirror, the distance between the mirrors and the image field, and the type of lens used in the laser for focusing the laser beam.

The following figure shows the basic projection system layout.

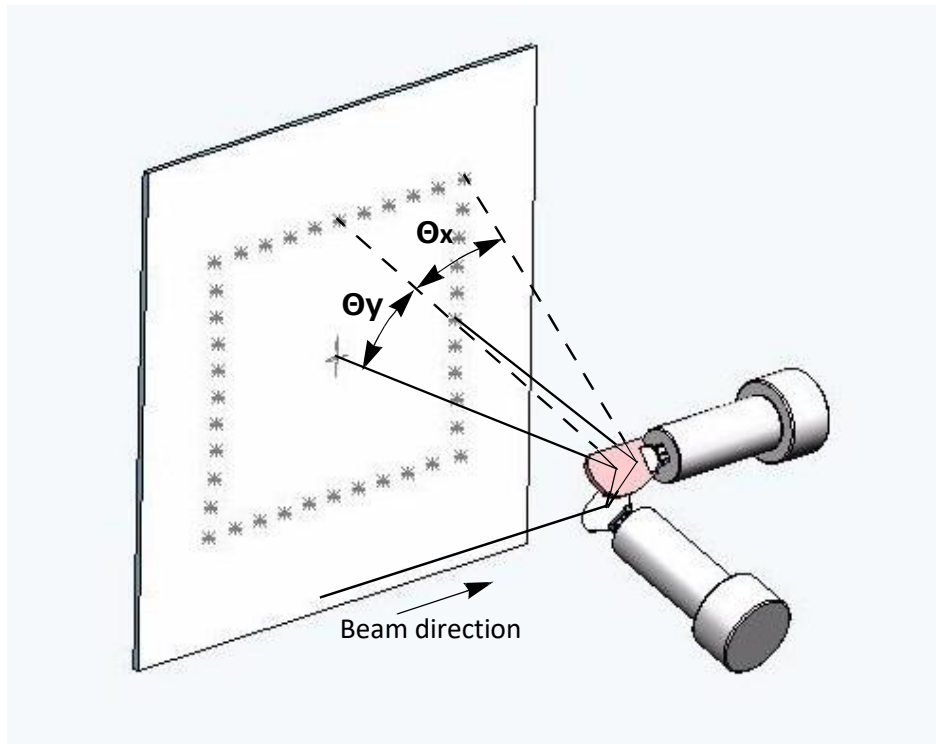


Figure 22 - PROJECTION SYSTEM LAYOUT

8.2.1 X-Y MIRROR INDUCED DISTORTION

Projection of a laser beam via an X-Y mirror set controlled by galvanometers induces distortion in the X-axis proportional to the tangent of the angle of the Y-axis mirror and the distance from the focal plane to the center of the Y-axis mirror. This distortion is also known as “pincushion” distortion.

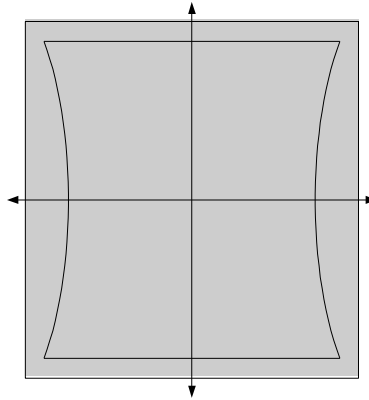


Figure 23 - PINCUSHION DISTORTION CAUSED BY AN X-Y MIRROR SET

8.2.2 F-THETA OBJECTIVE INDUCED DISTORTION

The addition of an F-theta objective in the laser field provides direct proportionality between the scan angle and the distance in the image field. The addition of an F-theta objective in the laser field also ensures that the focus lies on a flat surface. F-theta objective lenses, like all optical lenses, are not perfect and induce their own projection field distortions. This distortion, illustrated in the following figure, is called “pillow” distortion for what it does to a square image. In reality, this distortion is radially symmetric from the image field origin and can often be modeled as a third order polynomial. Many projection lens vendors will provide these model coefficients, or measurement data from which these coefficients can be derived. For many applications, however, this distortion is negligible.

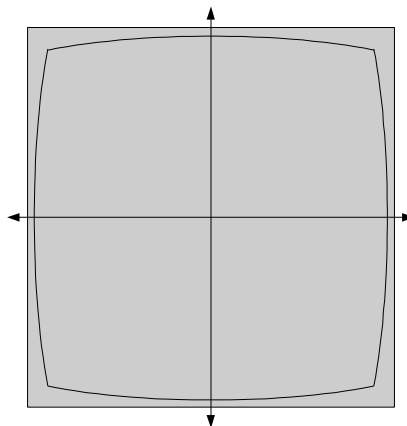


Figure 24 - PILLOW DISTORTION CAUSED BY F-THETA LENS

8.2.3 COMPOSITE DISTORTION AND CORRECTION METHODOLOGY

The two distortion components described above combine to create a distorted image field similar to that shown in the following figure. The SMC automatically compensates for this distortion by the use of correction tables.

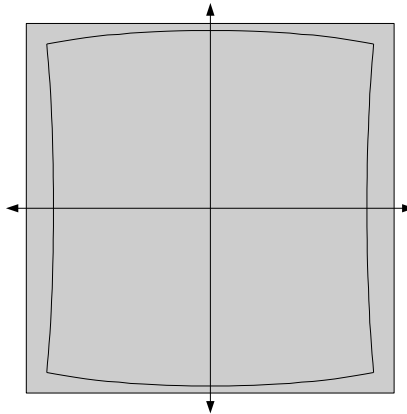


Figure 25 - COMPOSITE IMAGE FIELD DISTORTION

Correction tables represent a 65x65 element grid covering the full addressable projection range of the system. Each grid element contains three correction components: one each for the X, Y and Z axes. The components represent an offset that, if added to an ideal position command for that point, would alter the galvo positions such that the resulting projected point would fall onto a “perfect” grid (i.e., the point would be “corrected”).

During the micro-vectoring process at each update interval, the SMC calculates the ideal position of the mirrors along the path. It compares this value to the correction table grid and accesses the four grid points that immediately surround the calculated point. The corrections at these four points are proportionally averaged depending on how close the ideal point is to each grid point. This process, called bi-linear interpolation, produces a correction that is applied to the ideal point, and the result is then sent to the system D/A converters and serial digital command outputs.

8.2.4 MULTIPLE CORRECTION TABLE SUPPORT

The SMC has integral support for up to four independent three-axis correction tables. These tables are organized in pairs where the first pair is applied to the Auxiliary XY2-100 port and GSBUS axes 0, 1, 2, and the second pair is applied to the primary XY2-100 port and GSBUS axes 3, 4, 5. The job

parameter ActiveCorrectionTable dynamically selects which table of each pair that is actually used. The first of the two tables in the pair is intended to be used when actual laser processing is taking place. The second table of the pair is intended to be used with a pointer laser.

Table contents can be automatically loaded on board power-up from stored correction table files, or they can be dynamically loaded via the sendStreamData method of the session API.

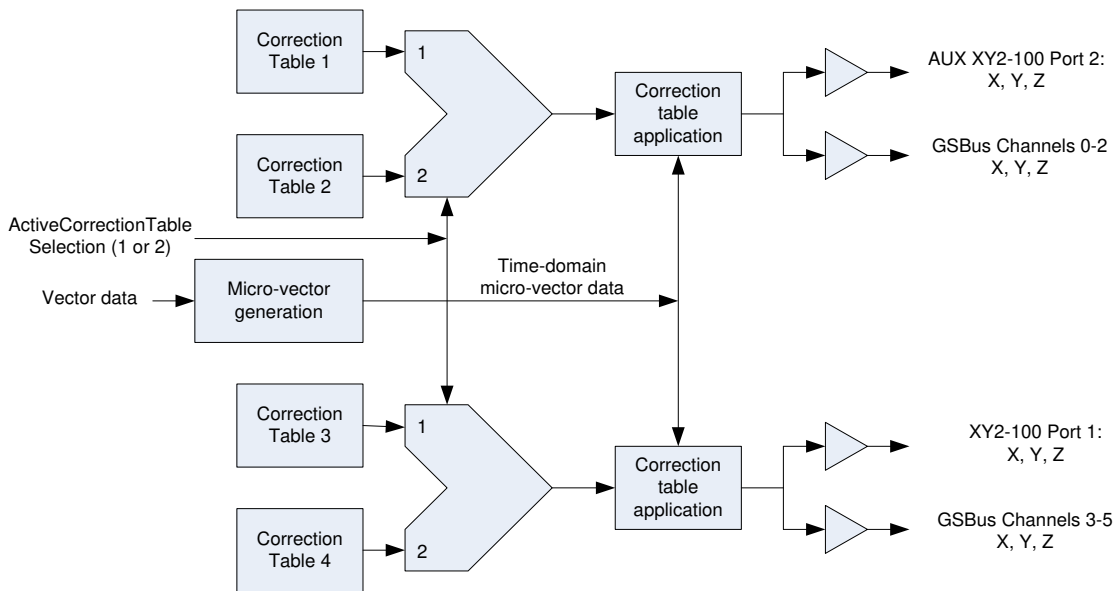


Figure 26 - MULTIPLE CORRECTION TABLE USAGE IN THE SMC

8.3 LASER TIMING CONTROL

The SMC provides very flexible laser control capability that is synchronized with galvo motion control. Six dedicated TTL-compatible signals (whose timing relationships are defined by the diagram below) are provided at all times. Not all signals may be required for a given customer laser configuration. An integrator need only select an appropriate subset of these signals, and configure them via software with appropriate timing parameters. Provisions are made for the synchronous control of two separate lasers running with two independent pulse-widths during the laser-on period. Laser control timing is specified in terms of laser timing “ticks” which can be set via software to an interval as small as 20ns to as large as 1.3ms with a resolution of 20ns. The typical tick value is set to 1μsec.

Appendix A - Theory of Operation

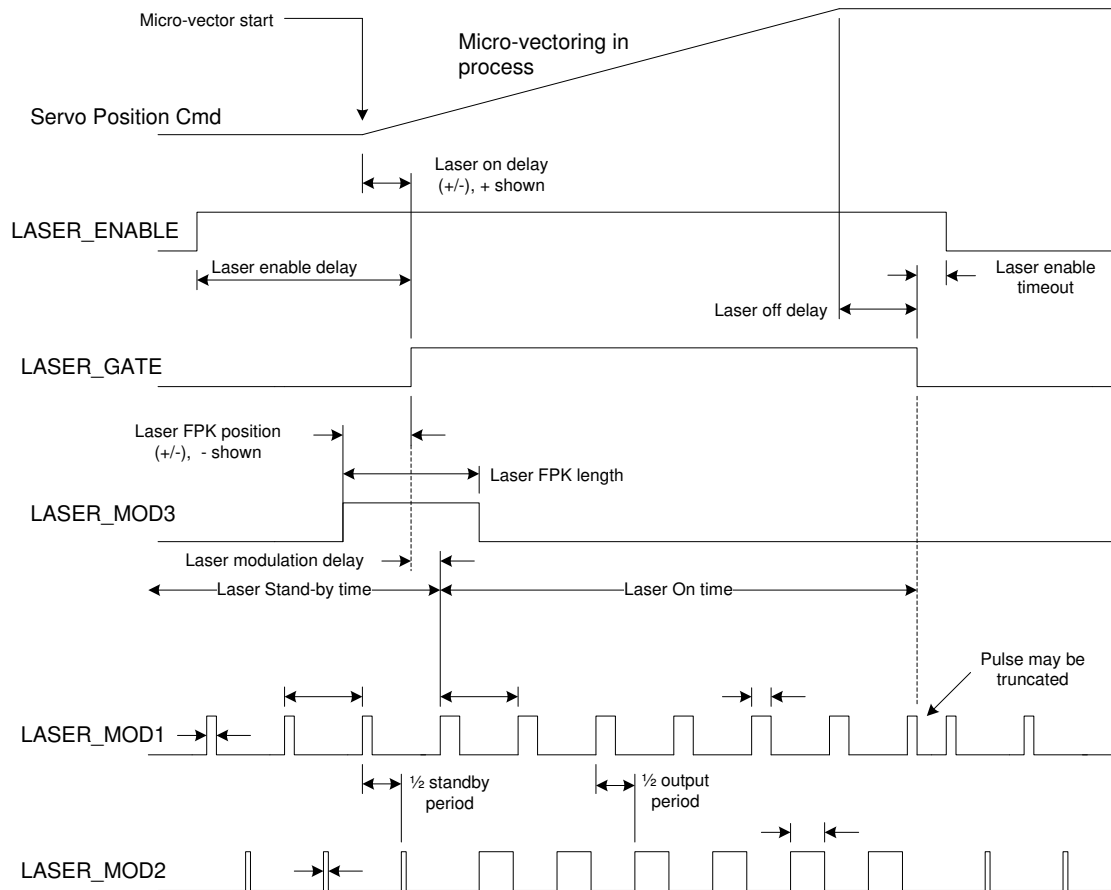


Figure 27 - LASER TIMING RELATIONSHIPS

Notes:

1. LaserEnableDelay, LaserEnableTimeout, and LaserModDelay must be ≥ 0 .
2. Laser Enable delay is relative to the leading edge of LASER_GATE, but the leading edge of LASER_ENABLE will never occur after any of the following:
 - Micro-vector start
 - The leading edge of LASER_GATE
 - The leading edge of LASER_MOD3 (FPK usage)
3. Laser On delay may be positive or negative and is relative to Micro-vector start.
4. LASER_MOD3 (FPK) position may be positive or negative and is relative to the leading edge of LASER_GATE.
5. Laser pulse generation starts relative to, but no earlier than, the leading edge of LASER_GATE or the leading edge of LASER_MOD3.

6. Standby pulse suppression is accomplished by setting the standby pulse width to zero.
7. The first laser-on laser pulse on LASER_MOD1 & 2 is always a full pulse.
8. The signal LASER_POINTER is also provided with multiple programmable functions to support pointer laser operation.

Figure 27 - Laser Timing Relationships on page 294 introduces 12 timing parameters that can be set to yield signal relationships that are suitable for controlling all known commercial lasers used in marking or projection scanning systems. The reference point for the timing is the beginning of micro-vectoring and is shown on the diagram as Micro-vector start.

When the marking engine processor encounters a mark instruction, it asserts the LASER_ENABLE signal and waits for the specified Laser Enable delay. The LASER_ENABLE signal is normally used to precondition fiber laser systems in anticipation of being called into action during a marking operation. LASER_ENABLE will remain asserted until the Laser Enable timeout period expires after marking has stopped, i.e. after the last vector of a sequence of marking vectors. If a new series of marking vectors begins before the Laser Enable timeout expires, LASER_ENABLE remains asserted and a new timeout period is armed.

When the Laser Enable delay expires, one of three things will happen based on the setting of the delay parameters:

1. Micro-vectoring begins if Laser On Delay and Laser First Pulse Killer (FPK) position are both positive.
2. LASER_GATE is asserted if Laser On Delay is negative and Laser FPK position is positive.
3. LASER_MOD3 is asserted if Laser FPK delay is negative and Laser On delay is also negative OR if Laser FPK delay is negative and the absolute value of Laser FPK delay is larger than Laser On delay if Laser On delay is positive.

As can be seen from *Figure 27 - Laser Timing Relationships* on page 294, the timing of laser emission is directly related to the timing of the LASER_GATE signal. Pulse emission will never occur earlier than the leading edge of LASER_GATE or LASER_MOD3, but it may be delayed after the leading edge of LASER_GATE by setting the Laser Modulation delay to a non-zero value. The LASER_MOD3 signal may be asserted any time before or after the leading edge of LASER_GATE. The signals LASER_MOD3 and LASER_MODn are dependently related to the timing of LASER_GATE. That is, if Laser On delay is changed, the system timing is changed to keep all three signals in the proper timing relationship.

The LASER_MOD1 and LASER_MOD2 signals are time-related in that the periods of the signals must be the same for the standby (laser not active) and output active (laser emitting) intervals. The phase of the two signals is programmable and is typically set to be 180 degrees apart from each other to ensure that the two lasers never fire at the same instant of time, thus reducing peak power

demands and reducing EMI effects. Otherwise, the pulse widths during the standby and output active intervals are independent and programmable for each signal.

4. The lasers are turned off automatically after the micro-vectoring completes and the Laser Off delay expires. The LASER_GATE signal is de-asserted and the LASER_MOD1/2 signals switch to the standby mode

8.4 SOFTWARE CONTROL OF LASER TIMING

The laser timing configuration is statically specified in an XML based configuration file stored on the SMC and is automatically applied at system boot-up. The configuration can be changed by reading it through the software Application Programming Interface (API), altering it, and then sending it back to the controller. Changes made this way would be applied every time the SMC re-initializes. The configuration information can also be specified dynamically in a job stream and applied on a temporary basis being persistent only until the next re-initialization. These concepts are described more fully in *Table 36 - Example IPG Fiber Laser Configuration XML* on page 313.

All of the programmable control elements of the SMC are manipulated through XML language constructs passed through the API. At system boot-up, XML configuration files are read from Flash memory on the controller and some of the parameters are applied to the hardware to pre-configure it. The Laser Configuration fixed-data contains definitions to specify laser marking and idle-time pulse-widths and frequency, signal polarities, FPK signal timing, etc. These parameters do not often change during a marking job, although provisions are made in the Job Stream XML specification to do so if required. Other laser timing parameters such as Laser On Delay and Laser Off Delay are expected to change as the job is tuned for best performance. These parameters are directly controlled by JobStream XML constructs, but not in the Laser Configuration XML specification.

Table 29 - LASER CONFIGURATION CONTROL XML EXAMPLES	
Static Configuration XML Example:	<code><LaserTiming>50</LaserTiming></code>
Dynamic Configuration XML Example:	<code><set id='LaserTiming'>50</set></code>
Example Description: Set the laser time base to 1μsec: $50 * 20\text{ns} = 1\mu\text{sec}$ "tick"	

Table 29 - LASER CONFIGURATION CONTROL XML EXAMPLES	
Static Configuration XML Example:	<code><LaserPipelineDelay>0</LaserPipelineDelay></code>
Dynamic Configuration XML Example:	<code><set id='LaserPipelineDelay'>0</set></code>
Example Description:	Normally zero except when using Cambridge Technology DC900, DC2000, or DC3000 digital servos. This value is used the delay all of the laser timing signals as a group relative to the galvo commands. The maximum pipeline delay value is equivalent to 4000 laser ticks so the specified value maximum will be reduces depending on the <u>LaserTiming</u> value. For example, if LaserTiming is 50 (1usec resolution) then the maximum value will be 4000usec. If LaserTiming is set to 5 (0.1usec resolution), then the maximum pipeline value is 400usec.
Static Configuration XML Example:	<code><'LaserPowerDelay'>1700</LaserPowerDelay></code>
Dynamic Configuration XML Example:	<code><set id='LaserPowerDelay'>1700</set></code>
Example Description:	The job will delay for 1.7msec every time the laser power is changed.
Static Configuration XML Examples:	<code><LaserModeConfig>0x0</LaserModeConfig></code>
Dynamic Configuration XML Example:	<code><set id='LaserModeConfig'>0x0</set></code>
Example Description:	<u>LaserModeConfig</u> uses a bit-mask to represent the various signal polarities.
Static Configuration XML Example:	<code><LaserEnableDelay>7</LaserEnableDelay></code>
Dynamic Configuration XML Example:	<code><set id='LaserEnableDelay'>7</set></code>
Example Description:	Wait 7msec after asserting the LASERENABLE signal.
Static Configuration XML Example:	<code><LaserEnableTimeout>50</LaserEnableTimeout></code>
Dynamic Configuration XML Example:	<code><set id='LaserEnableTimeout'>50</set></code>
Example Description:	Deassert LASERENABLE if there is no laser activity requested within 50msec of when the laser turned off.
Static Configuration XML Example:	<code><LaserModDelay>20</LaserModDelay></code>
Dynamic Configuration XML Example:	<code><set id='LaserModDelay'>20</set></code>
Example Description:	Delay the modulation of the laser for 20 laser timing ticks after LASER_GATE is asserted.

Table 29 - LASER CONFIGURATION CONTROL XML EXAMPLES	
Static Configuration XML Example:	<LaserFPK position='-30' width='10'/>
Dynamic Configuration XML Example:	<set id='LaserFPK'>-30; 10</set>
Example Description:	Assert LASER_MOD3 30 laser timing ticks in advance of the leading edge of LASER_GATE. Deassert LASER_MOD3 10 laser timing ticks after it was asserted.
Static Configuration XML Examples:	<LaserStandby laser='1' width='5' period='200'/> <LaserStandby laser='2' width='5' period='200'/>
Dynamic Configuration XML Example:	<set id='LaserStandby'>1; 5; 200</set> <set id='LaserStandby'>2; 5; 200</set>
Example Description:	For Lasers 1 & 2, set the stand-by (idle) pulse width to 5 laser timing ticks and set the period to 200 ticks. This is a pulse frequency of 5KHz provided that <i>LaserTiming</i> is set to 50.
Dynamic Configuration XML Example:	<set id='LaserOnDelay'>150</set>
Example Description:	LASER_GATE is asserted 150 laser timing ticks after the start of micro-vectoring.
Dynamic Configuration XML Example:	<set id='LaserOffDelay'>100</set>
Example Description:	LASER_GATE is deasserted 100 laser timing ticks after the micro-vectoring has completed.
Dynamic Configuration XML Example:	<set id='LaserPulse'>1; 8; 15</set>
Example Description:	For Laser 1, set the “Laser On” pulse width to 8 laser timing ticks and set the period to 15 ticks. This is a pulse frequency of 66.7KHz provided that <i>LaserTiming</i> is set to 50.
Dynamic Configuration XML Example:	<set id='LaserPulse'>2; 10; 5</set>
Example Description:	Laser 2 period always follows Laser 1. For Laser 2, set “Laser On” pulse width 10 ticks and the phase shift to 5 ticks. This is a pulse frequency of 66.7KHz provided that <i>LaserTiming</i> is set to 50.

8.4.1 LASER TIMING EMULATION

Traditional laser scanning controllers often use fixed signal sets and constrained timing relationships to provide laser control, whereas the SMC uses a completely flexible and programmable suite of signals. The SMC can be configured to emulate the timing produced by other commercial controllers because of the flexible nature of the laser timing generator.

Typical laser configurations are shown in the following diagrams. These configurations emulate the laser control performed by the RAYLASE AG SP-ICE card, and SCANLAB RTC3/4/5 and SCANAlone series of scan head controllers. These configurations are by no means the only ones possible, and new laser systems are frequently introduced. Most notably, fiber lasers have become much more reliable and affordable, offering compact packaging and highly efficient energy properties. The SMC has been specifically designed to accommodate the unique timing requirements of these lasers.

Along with each diagram are static and dynamic XML examples for configuring the laser. Only those parameters that are meaningful for the illustration are specified in the examples. Other parameters—such as those used to set signal polarities, Laser Enable Delay/Timeout, Standby (Tickle) timing, Laser Power Delay and Laser Pipeline Delay—are almost always set to pre-defined values. Laser Pulse timing, although potentially variable during a job, does not affect the fundamental signal relationships that define the laser emulation modes. In addition, the specification of a laser timing “tick” is most conveniently set to a 1µsec interval, which is assumed in the examples.

CO₂ Laser Timing

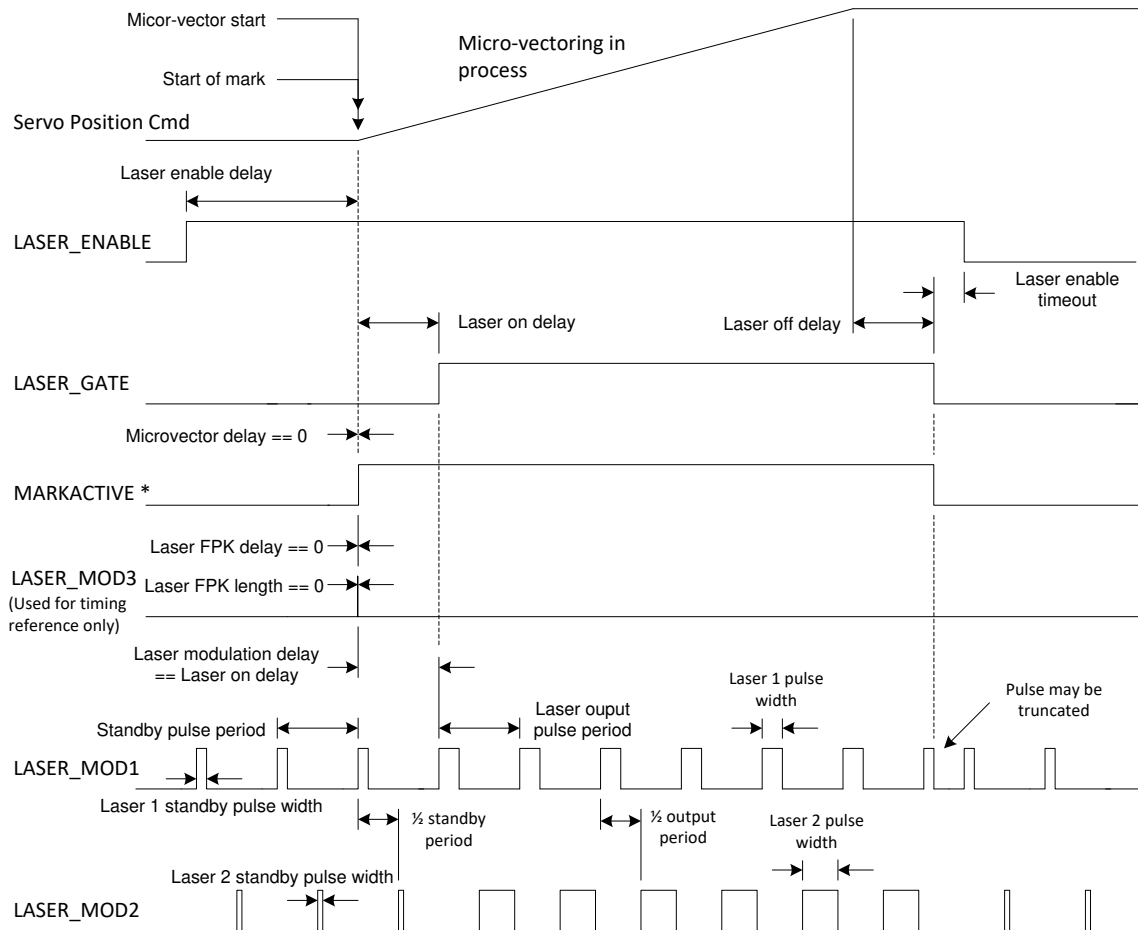


Figure 28 - LASER TIMING FOR CO₂ LASER SYSTEMS

The simplest emulation mode is for CO₂ lasers. These lasers do not require a Laser FPK signal so these parameters are set to zero. LASER_ENABLE is also not typically needed, therefore the Laser Enable delay and Laser Enable timeout can be set to zero to maximize throughput. In fact, whenever LASER_ENABLE is not required, the Laser Enable delay should be set to zero.

Table 30 - EXAMPLE CO₂ LASER CONFIGURATION XML

Static Configuration XML	Dynamic Configuration XML	Example Description
<LaserEnableDelay> 0 </LaserEnableDelay >	<set id='LaserEnableDelay'> 0 </set>	Maximizes throughput
<LaserEnableTimeout> 0 </LaserEnableTimeout >	<set id='LaserEnableTimeout'> 0 </set>	Maximizes throughput
<LaserModDelay> 0 </LaserModDelay>	<set id='LaserModDelay'> 0 </set>	No modulation delay required
<LaserFPK> 0, 0 </LaserFPK>	<set id='LaserFPK ' > 0, 0 </set>	No FPK required
<LaserStandby> 1; 5; 200 </LaserStandby>	<set id='LaserStandby'> 1; 5; 200 </set>	Laser 1 stand-by; pulse width = 5 laser timing ticks (5μsec); pulse period = 200 ticks (200μsec) = 5KHz
<LaserStandby> 2; 10; 200 </LaserStandby>	<set id='LaserStandby'> 2; 10; 200 </set>	Laser 2; pulse width = 10 laser timing ticks (10μsec); pulse period = 200 ticks (200μsec) = 5KHz, must be same as Laser 1
N/A	<set id='LaserOnDelay'> 150 </set>	150 laser timing ticks = 150μsec
N/A	<set id='LaserOffDelay'> 100 </set>	100 laser timing ticks = 100μsec
N/A	<set id='LaserPulse'> 1; 8; 15 </set>	Laser 1 operating; pulse width = 8 laser timing ticks (8μsec); pulse period = 15 ticks (15μsec) = 66.7KHz
N/A	<set id='LaserPulse'> 2; 10; 15 </set>	Laser 2 operating; pulse width = 10 laser timing ticks (10μsec); pulse period = 15 ticks (15μsec) = 66.7KHz, must be same as Laser 1

Nd:YAG Emulation Mode-1 Timing

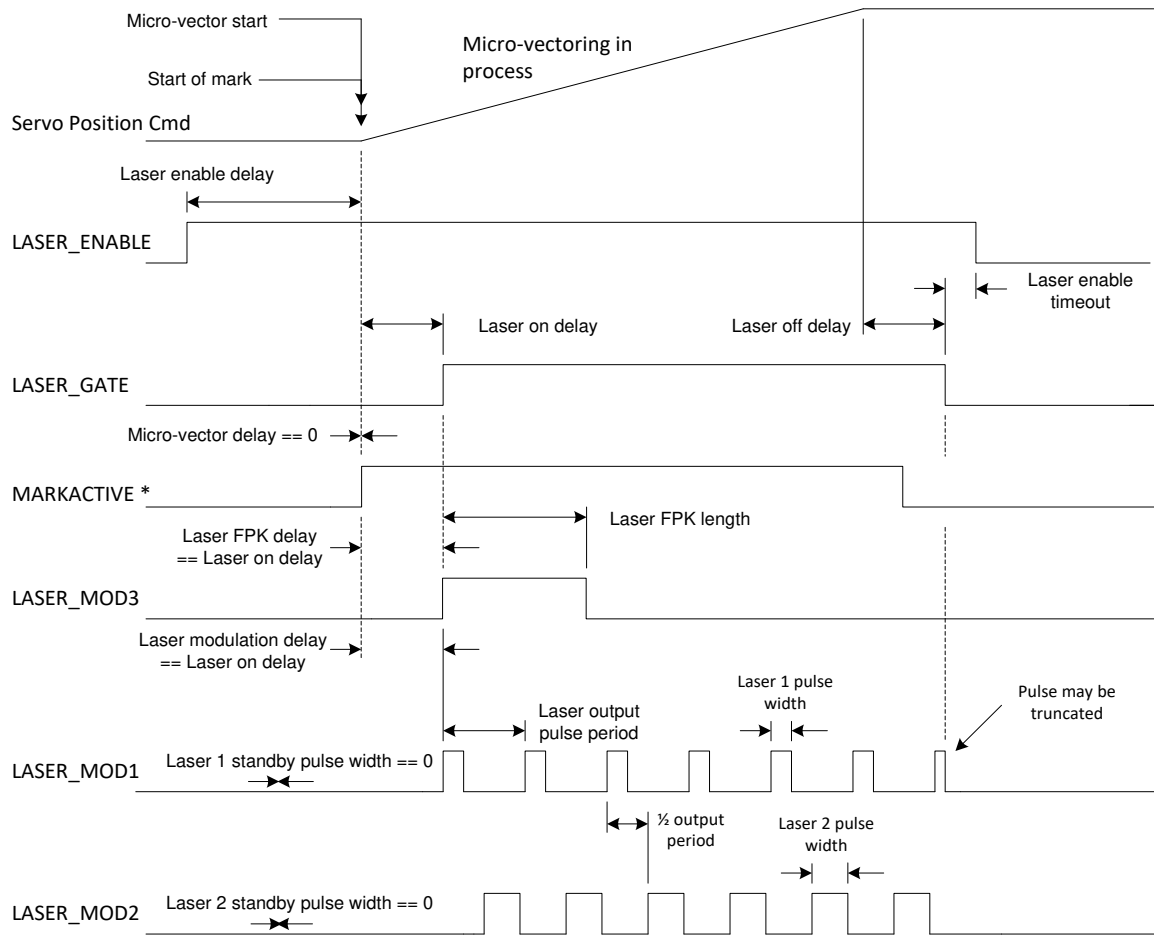


Figure 29 - Nd:YAG EMULATION MODE-1 (RAYLASE Nd:YAG MODE-1 AND SCANLAB YAG 1)

Most of the YAG modes do not require standby or idle pulses. To suppress these pulses, the Standby pulse width and pulse period are set to zero. In this mode, the LASER_MOD3 is asserted coincident with the LASER_GATE and LASER_MOD1 signals, but its assertion can have variable length. If the Laser On delay is modified, the timing of LASER_MOD3 and LASER_MOD1 track with it.

Table 31 - EXAMPLE ND:YAG MODE-1 LASER CONFIGURATION XML

Static Configuration XML	Dynamic Configuration XML	Example Description
<LaserEnableDelay> 0 </LaserEnableDelay >	<set id='LaserEnableDelay'> 0 </set>	Maximizes throughput
<LaserEnableTimeout> 0 </LaserEnableTimeout >	<set id='LaserEnableTimeout'> 0 </set>	Maximizes throughput
<LaserModDelay> 0 </LaserModDelay>	<set id='LaserModDelay'> 0 </set>	No modulation delay required
<LaserFPK> 0, 15 </LaserFPK>	<set id='LaserFPK'> 0, 15 </set>	Example FPK length set to 15usec with no shift
<LaserStandby> 1; 0; 0 </LaserStandby>	<set id='LaserStandby'> 1; 0; 0 </set>	1 = laser; no tickle pulses required
<LaserStandby> 2; 0; 0 </LaserStandby>	<set id='LaserStandby'> 2; 0; 0 </set>	2 = laser; no tickle pulses required
N/A	<set id='LaserOnDelay'> 150 </set>	150 laser timing ticks = 150μsec
N/A	<set id='LaserOffDelay'> 100 </set>	100 laser timing ticks = 100μsec
N/A	<set id='LaserPulse'> 1; 8; 15 </set>	Laser 1 operating; pulse width = 8 laser timing ticks (8μsec); pulse period = 15 ticks (15μsec) = 66.7KHz
N/A	<set id='LaserPulse'> 2; 10; 15 </set>	Laser 2 operating; pulse width = 10 laser timing ticks (10μsec); pulse period = 15 ticks (15μsec) = 66.7KHz

Nd:YAG Emulation Mode-2 Timing

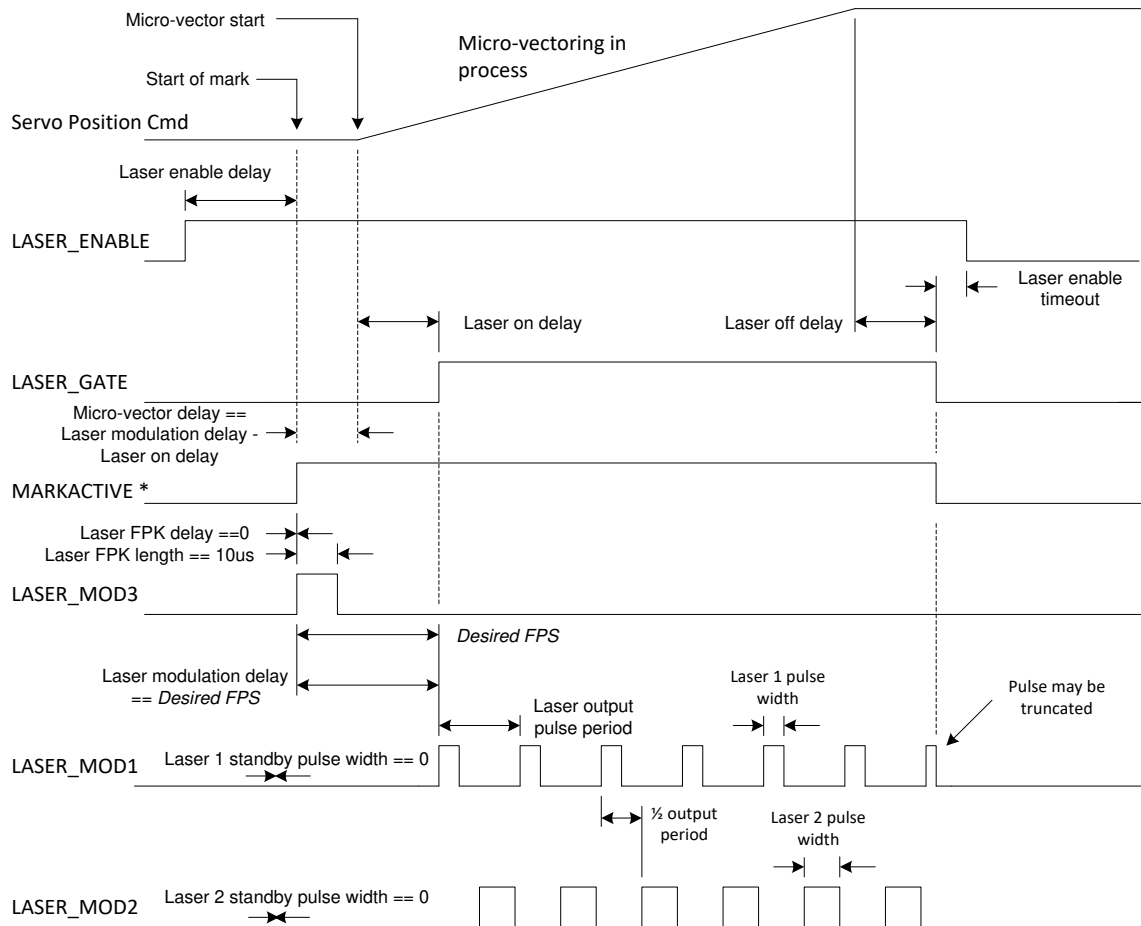


Figure 30 - Nd:YAG EMULATION MODE-2 (RAYLASE Nd:YAG MODE-2)

In this mode, the LASER_MOD3 signal is a 10μ sec pulse asserted a variable amount of time prior to the assertion of LASER_GATE and the coincident generation of pulses. This timing is typically suited for Lee and Coherent lasers.

Table 32 - EXAMPLE ND:YAG EMULATION MODE-2 LASER CONFIGURATION XML

Static Configuration XML	Dynamic Configuration XML	Example Description
<LaserEnableDelay> 0 </LaserEnableDelay >	<set id='LaserEnableDelay'> 0 </set>	Maximizes throughput
<LaserEnableTimeout> 0 </LaserEnableTimeout >	<set id='LaserEnableTimeout'> 0 </set>	Maximizes throughput
<LaserModDelay> 0 </LaserModDelay>	<set id='LaserModDelay'> 0 </set>	No modulation delay required
<LaserFPK> -20, 10 </LaserFPK>	<set id='LaserFPK '> -20, 10 </set>	Example FPK length set to 10μsec with a minus 20μsec shift relative to LASER_GATE
<LaserStandby> 1; 0; 0 </LaserStandby>	<set id='LaserStandby'> 1; 0; 0 </set>	1 = laser; no tickle pulses required
<LaserStandby> 2; 0; 0 </LaserStandby>	<set id='LaserStandby'> 2; 0; 0 </set>	2 = laser; no tickle pulses required
N/A	<set id='LaserOnDelay'> 150 </set>	150 laser timing ticks = 150μsec
N/A	<set id='LaserOffDelay'> 100 </set>	100 laser timing ticks = 100μsec
N/A	<set id='LaserPulse'> 1; 8; 15 </set>	Laser 1 operating; pulse width = 8 laser timing ticks (8μsec); pulse period = 15 ticks (15μsec) = 66.7KHz
N/A	<set id='LaserPulse'> 2; 10; 15 </set>	Laser 2 operating; pulse width = 10 laser timing ticks (10μsec); pulse period = 15 ticks (15μsec) = 66.7KHz

Nd:YAG Emulation Mode-3 Timing

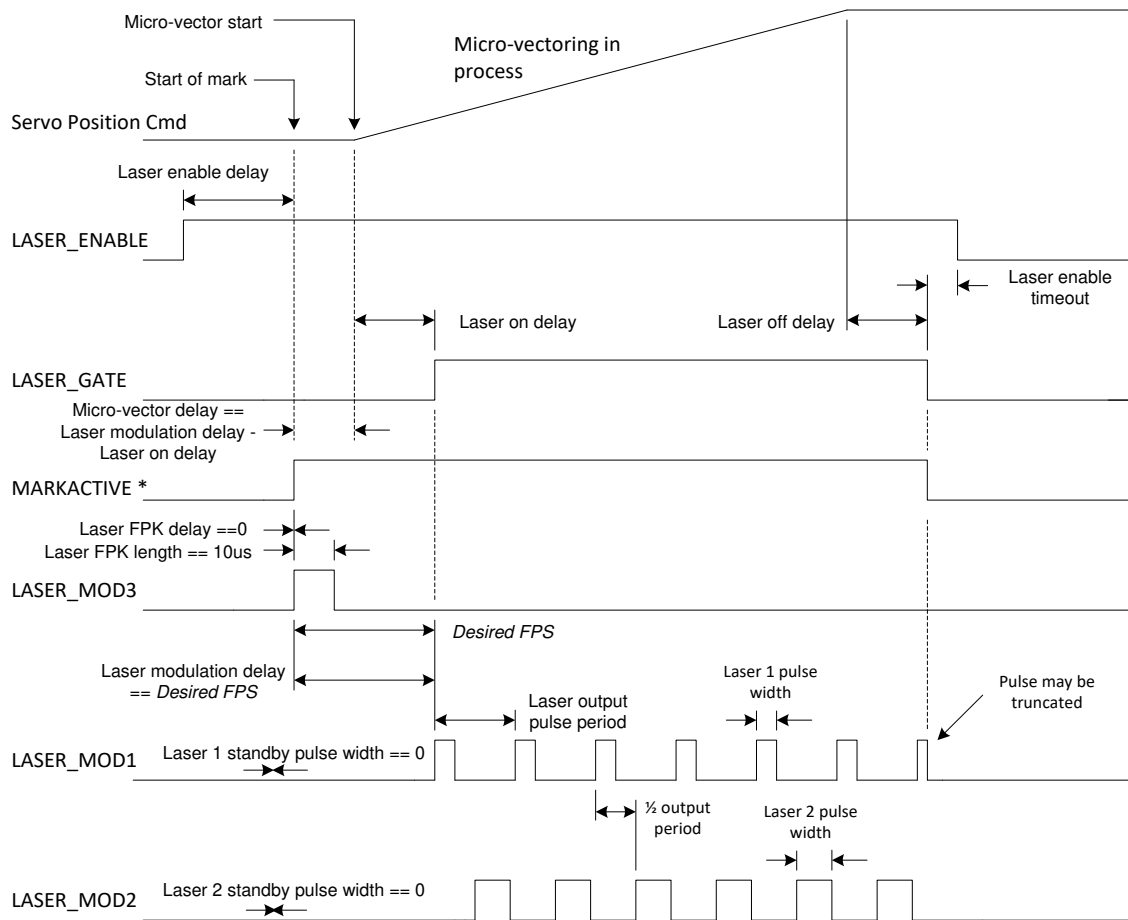


Figure 31 - Nd:YAG EMULATION MODE-3 (RAYLASE Nd:YAG MODE-3)

This mode is very similar to Mode-2. The difference is that Laser FPK length can vary. Spectron lasers normally use this type of timing.

Table 33 - EXAMPLE Nd:YAG MODE-3 LASER CONFIGURATION XML

Static Configuration XML	Dynamic Configuration XML	Example Description
<code><LaserEnableDelay>0</LaserEnableDelay></code>	<code><set id='LaserEnableDelay'>0</set></code>	Maximizes throughput

Table 33 - EXAMPLE ND:YAG MODE-3 LASER CONFIGURATION XML

Static Configuration XML	Dynamic Configuration XML	Example Description
<LaserEnableTimeout> 0 </LaserEnableTimeout >	<set id='LaserEnableTimeout'> 0 </set>	Maximizes throughput
<LaserModDelay> 0 </LaserModDelay>	<set id='LaserModDelay'> 0 </set>	No modulation delay required
<LaserFPK> -20, 18 </LaserFPK>	<set id='LaserFPK ' > -20, 18 </set>	Example FPK length set to 18μsec with a minus 20μsec shift relative to LASER_GATE
<LaserStandby> 1; 0; 0 </LaserStandby>	<set id='LaserStandby'> 1; 0; 0 </set>	1 = laser; no tickle pulses required
<LaserStandby> 2; 0; 0 </LaserStandby>	<set id='LaserStandby'> 2; 0; 0 </set>	2 = laser; no tickle pulses required
N/A	<set id='LaserOnDelay'> 150 </set>	150 laser timing ticks = 150μsec
N/A	<set id='LaserOffDelay'> 100 </set>	100 laser timing ticks = 100μsec
N/A	<set id='LaserPulse'> 1; 8; 15 </set>	Laser 1 operating; pulse width = 8 laser timing ticks (8μsec); pulse period = 15 ticks (15μsec) = 66.7KHz
N/A	<set id='LaserPulse'> 2; 10; 15 </set>	Laser 2 operating; pulse width = 10 laser timing ticks (10μsec); pulse period = 15 ticks (15μsec) = 66.7KHz

Nd:YAG Emulation Mode-4 Timing

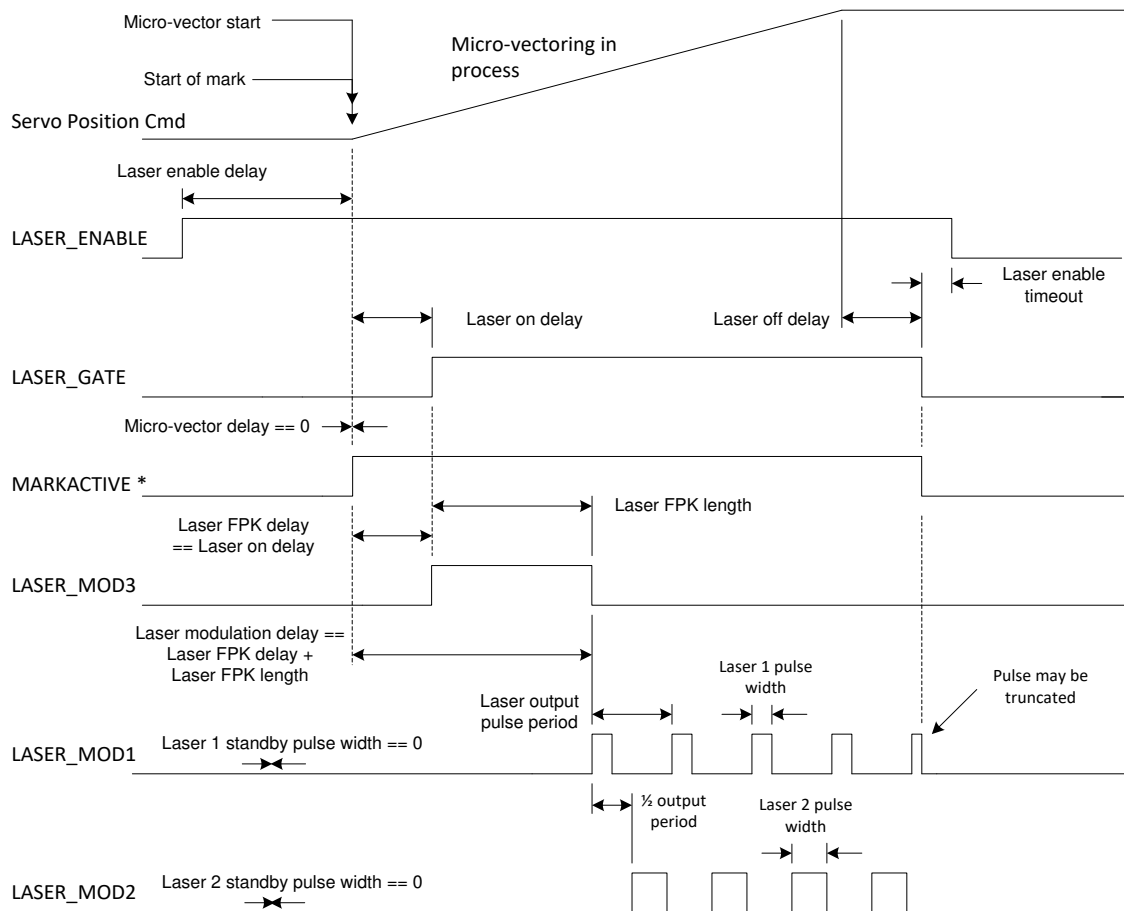


Figure 32 - ND:YAG EMULATION MODE-4 (SCANLAB YAG 2)

In this mode, the LASE_MOD3 signal leading edge is coincident with the leading edge of LASER_GATE, and the generation of the laser pulses is delayed to be coincident with the trailing edge of the LASER_MOD3 signal.

Table 34 - EXAMPLE ND:YAG MODE-4 LASER CONFIGURATION XML

Static Configuration XML	Dynamic Configuration XML	Example Description
<LaserEnableDelay> 0 </LaserEnableDelay >	<set id='LaserEnableDelay'> 0 </set>	Maximizes throughput

Table 34 - EXAMPLE ND:YAG MODE-4 LASER CONFIGURATION XML

Static Configuration XML	Dynamic Configuration XML	Example Description
<LaserEnableTimeout> 0 </LaserEnableTimeout >	<set id='LaserEnableTimeout'> 0 </set>	Maximizes throughput
<LaserModDelay> 15 </LaserModDelay>	<set id='LaserModDelay'> 15 </set>	Laser modulation delayed by the same value as the LASER_MOD3 length
<LaserFPK> 0, 15 </LaserFPK>	<set id='LaserFPK '> 0, 15 </set>	Example FPK length set to 15µsec with no shift relative to LASER_GATE
<LaserStandby> 1; 0; 0 </LaserStandby>	<set id='LaserStandby'> 1; 0; 0 </set>	1 = laser; no tickle pulses required
<LaserStandby> 2; 0; 0 </LaserStandby>	<set id='LaserStandby'> 2; 0; 0 </set>	2 = laser; no tickle pulses required
N/A	<set id='LaserOnDelay'> 150 </set>	150 laser timing ticks = 150µsec
N/A	<set id='LaserOffDelay'> 100 </set>	100 laser timing ticks = 100µsec
N/A	<set id='LaserPulse'> 1; 8; 15 </set>	Laser 1 operating; pulse width = 8 laser timing ticks (8µsec); pulse period = 15 ticks (15µsec) = 66.7KHz
N/A	<set id='LaserPulse'> 2; 10; 15 </set>	Laser 2 operating; pulse width = 10 laser timing ticks (10µsec); pulse period = 15 ticks (15µsec) = 66.7KHz, must be same as Laser 1

Nd:YAG Emulation Mode-5 Timing

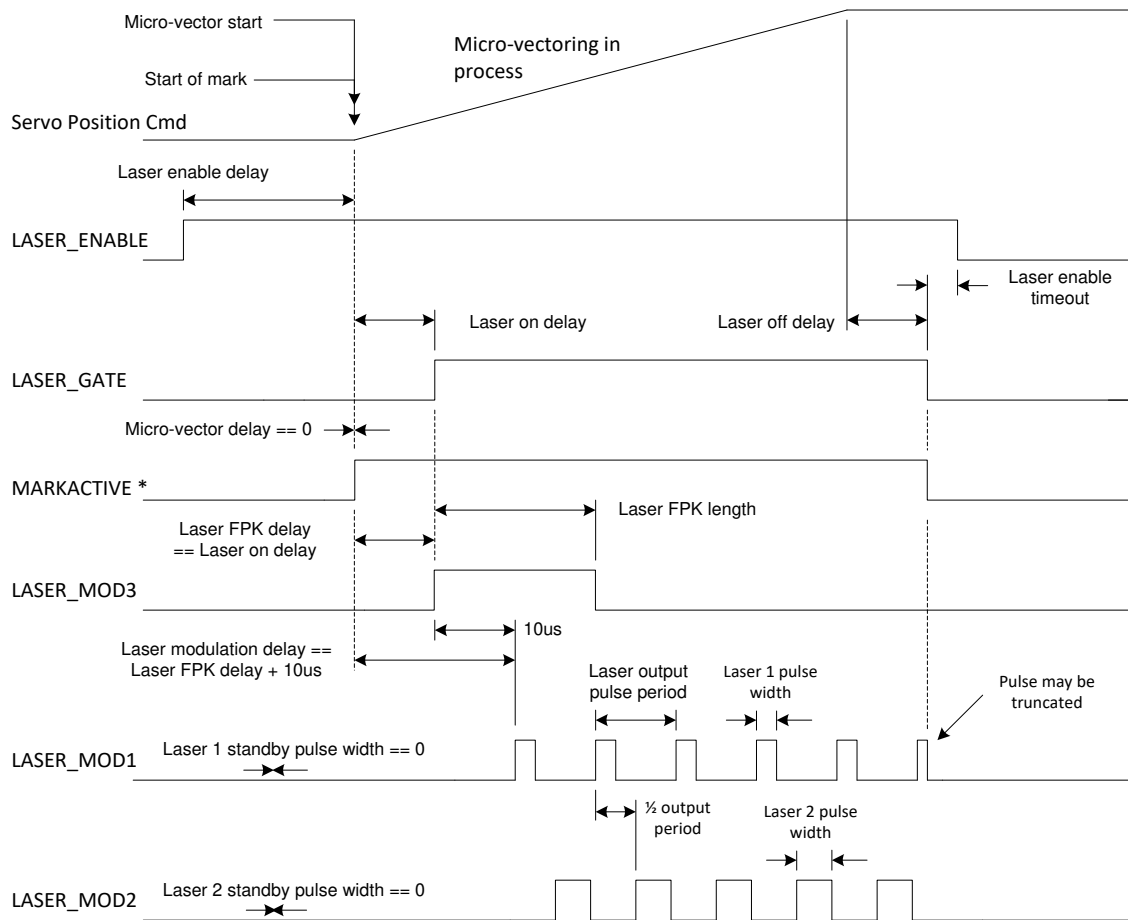


Figure 33 - Nd:YAG EMULATION MODE-5 (SCANLAB YAG 3)

This mode is very similar to emulation mode-4. The difference is that the start of laser pulse generation is 10µ sec after the coincident leading edges of LASER_GATE and LASER_MOD3.

Table 35 - EXAMPLE Nd:YAG MODE-5 LASER CONFIGURATION XML

Static Configuration XML	Dynamic Configuration XML	Example Description
<code><LaserEnableDelay>0</LaserEnableDelay ></code>	<code><set id='LaserEnableDelay'>0</set></code>	Maximizes throughput

Table 35 - EXAMPLE ND:YAG MODE-5 LASER CONFIGURATION XML

Static Configuration XML	Dynamic Configuration XML	Example Description
<code><LaserEnableTimeout>0</LaserEnableTimeout></code>	<code><set id='LaserEnableTimeout'>0</set></code>	Maximizes throughput
<code><LaserModDelay>10</LaserModDelay></code>	<code><set id='LaserModDelay'>10</set></code>	Laser modulation delayed by 10μsec relative to LASER_GATE
<code><LaserFPK>0, 20</LaserFPK></code>	<code><set id='LaserFPK'>0, 20</set></code>	Example FPK length set to 20μsec with no shift relative to LASER_GATE
<code><LaserStandby>1; 0; 0</LaserStandby></code>	<code><set id='LaserStandby'>1; 0; 0</set></code>	1 = laser; no tickle pulses required
<code><LaserStandby>2; 0; 0</LaserStandby></code>	<code><set id='LaserStandby'>2; 0; 0</set></code>	2 = laser; no tickle pulses required
N/A	<code><set id='LaserOnDelay'>150</set></code>	150 laser timing ticks = 150μsec
N/A	<code><set id='LaserOffDelay'>100</set></code>	100 laser timing ticks = 100μsec
N/A	<code><set id='LaserPulse'>1; 8; 15</set></code>	Laser 1 operating; pulse width = 8 laser timing ticks (8μsec); pulse period = 15 ticks (15μsec) = 66.7KHz
N/A	<code><set id='LaserPulse'>2; 10; 15</set></code>	Laser 2 operating; pulse width = 10 laser timing ticks (10μsec); pulse period = 15 ticks (15μsec) = 66.7KHz

Fiber Laser Timing

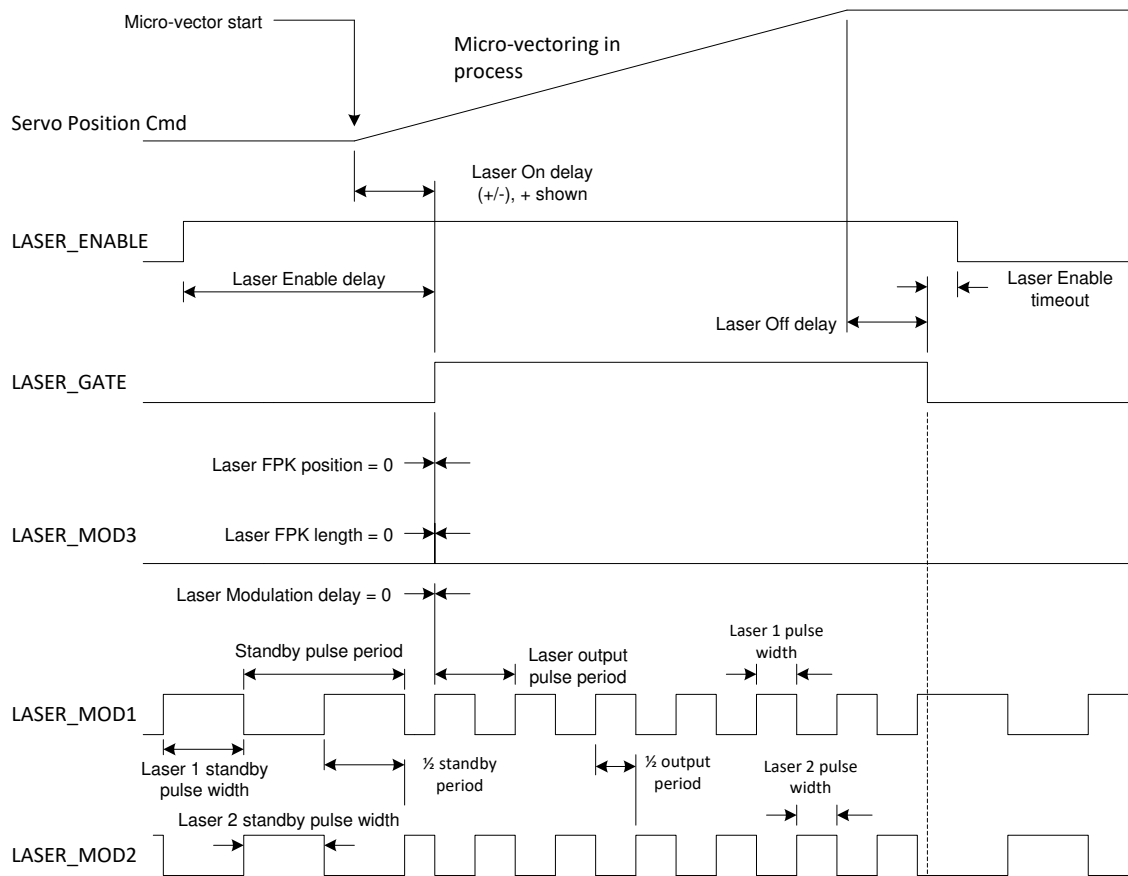


Figure 34 - FIBER LASER TIMING

Pulsed fiber lasers have recently become very popular because of a reduced cost of ownership relative to more traditional YAG lasers. The IPG YLP series of lasers introduces a new control signal requirement that is met with the LASER_ENABLE signal of the SMC. The MO (Master Oscillator) signal defined in the IPG “B” interface specification is intended to be driven by the LASER_ENABLE signal of the SMC. This signal is used to prepare the fiber laser to generate output pulses and must be asserted at least 7ms before pulses are required. In addition, this signal should be deasserted after laser emission in order to save power and extend the life of the laser. Deassertion, however, should not be done too quickly in order to avoid the overhead of restarting the laser. Deassertion is usually done after all marking is done in a job. In the case of the SMC, a timeout is provided to automatically deassert the LASER_ENABLE signal after a period of inactivity.

In the above diagram notice that the LASER_MOD3 signal is made inactive (i.e., it is not required by the interface.) The pulse width of the standby and active periods is set to 50% of the pulse period (square wave) since laser emission is triggered on the leading edge of the pulse. Pulse width does not determine the level of power emitted; only the pulse frequency (or period) determines average power. In practice, the pulse-width-to-period ratio can be in a range of 0.1 to 0.9.



CAUTION

The IPG laser Type A interface specifies that the pulse period must not be longer than a minimum value. The SMC does *not* protect against incorrect programming; the application must prevent incorrect values from being used.

Table 36 - EXAMPLE IPG FIBER LASER CONFIGURATION XML

Static Configuration XML	Dynamic Configuration XML	Example Description
<code><LaserEnableDelay>8</LaserEnableDelay></code>	<code><set id='LaserEnableDelay'>8</set></code>	Minimum master oscillator startup time
<code><LaserEnableTimeout>50</LaserEnableTimeout></code>	<code><set id='LaserEnableTimeout'>50</set></code>	Shut down laser master oscillator if no laser activity for 10msec
<code><LaserModDelay>0</LaserModDelay></code>	<code><set id='LaserModDelay'>0</set></code>	No modulation delay required
<code><LaserFPK>0, 0</LaserFPK></code>	<code><set id='LaserFPK'>0, 0</set></code>	No FPK required
<code><LaserStandby>1; 25; 50</LaserStandby></code>	<code><set id='LaserStandby'>1; 25; 50</set></code>	Laser 1 stand-by; pulse width = 25 laser timing ticks (25µsec); pulse period = 50 ticks (50µsec) = 20.0KHz
<code><LaserStandby>2; 25; 50</LaserStandby></code>	<code><set id='LaserStandby'>1; 25; 50</set></code>	Laser 2; Settings the same as Laser 1
N/A	<code><set id='LaserOnDelay'>150</set></code>	150 laser timing ticks = 150µsec
N/A	<code><set id='LaserOffDelay'>100</set></code>	100 laser timing ticks = 100µsec

Table 36 - EXAMPLE IPG FIBER LASER CONFIGURATION XML

Static Configuration XML	Dynamic Configuration XML	Example Description
N/A	<code><set id='LaserPulse'>1; 8; 15</set></code>	Laser 1 operating; pulse width = 8 laser timing ticks (8μsec); pulse period = 15 ticks (15μsec) = 66.7KHz
N/A	<code><set id='LaserPulse'>2; 10; 15</set></code>	Laser 2 operating; pulse width = 10 laser timing ticks (10μsec); pulse period = 15 ticks (15μsec) = 66.7KHz

9 APPENDIX B - ERROR CODES

9.1 XML API ERROR CODES

Table 37 - API ERROR CODES

Table 38 -

Error Name	Code	Description
Success	0	Operation successful
Error_AccessDenied	1	TCP/IP networking access was denied
Error_Communications	2	TCP/IP network communications error occurred
Error_NotConnected	3	Client is not connected to the server
Error_IllegalClientId	4	Internal error
Error_InvalidPersistState	5	Internal error
Error_ServerNameNotFound	8	Requested server name is not valid
Error_InvalidParameter	9	Bad parameter to a method call
Error_Network	10	TCP/IP networking error
Error_DataNotFound	11	Requested data file not found
Error_PathNotFound	12	Specified path does not exist
Error_Access	13	Access to server file system was denied
Error_LocalAccess	14	Access to the client file system was denied or Server is under control of a local pendant

Table 37 - API ERROR CODES**Table 38 -**

Error Name	Code	Description
Error_DataUnknown	15	XML data type is unknown
Error_EventHandling	16	Internal event processing error
Error_NotAvailable	17	Server is not currently available
Error_CancelByUser	18	Server is aborting by user
Error_Aborting	19	Server is currently aborting
Error_LicenseUnavailable	21	License is not found
Error_LicenseAccessDenied	22	License is not granted
Error_Exception	23	Internal error
Error_Timeout	24	Requested action timed out
Error_NoData	25	The requested fixed data was empty
Error_DataExists	26	Destination file already exists and over-write not specified
Error_RemoteAccess	28	Server is already connected to a client
Error_StateError	29	Server is in an error state and unavailable
Error_NotFound	30	Server is not found
Error_Buffer_Full	31	Streaming data transmit buffer is full
Error_RemoteAPIUnavailable	32	Remote API is in use by another client
Error_RemoteAPITimeout	33	Timed out connecting to the remote API
Error_SocketException	34	Network socket error detected

Table 37 - API ERROR CODES

Table 38 -		
Error Name	Code	Description
Error_CommandSyntax	37	Command syntax error
Error_XMLJobSyntax	38	XML job syntax error
Error_Busy	39	System is busy in running a job
Error_LoginInProgress	99	Error during Login

9.2 REMOTE CONTROL ERROR CODES

Table 39 - REMOTE CONTROL RETURN CODES

Code	Meaning	Description
0	Success	Command processed with no error.
100	No Files Found	There were no job files in the device or folder.
101	No Drive	No USB disk drive was found.
105	JobException	Job causes exception during execution.
106	Not In Host Control	The command required that exclusive control of the SMC first be obtained by executing the <u>TakeHostControl</u> command.
108	Error Job Busy	The command cannot execute because a job is running.
110	Error Software	An internal software exception occurred.
111	Load Fail	Cannot load the job.
112	No Objects	Required objecta not exist.

Table 39 - REMOTE CONTROL RETURN CODES

Code	Meaning	Description
114	Job Files Format	The file type (file extension) is not a valid job.
121	File Not Found	The named job file was not found.
122	Idle	No job is running.
123	Busy	A job is running.
124	No Job	No job is loaded to run.
125	In Control	The Remote Control is in control as the host.
126	Not In Control	The Remote Control is not as the host.
127	Bad Command	The command was not recognized.
128	Bad Arg	The command passed inappropriately formed arguments (or no argument if an argument was required).
129	Arg Out of Range	The command passed argument was not in allowed range.
131	Abort Detected	Job is aborting or aborted.
132	Arg Count Not Matched	Wrong number of command arguments was passed.
202	Abort	Job was Aborted.
207	Cannot Open Port	Cannot open the serial port.
208	Port Not Open	The serial port was not opened before the requested command.
209	Port Timeout	The serial port timed out waiting for input.
210	Wrong Port Number	An invalid COM port ID was specified.

9.3 LastError Code Descriptions

Table 40 - LASTERROR CODE DESCRIPTIONS

Value	Description
0	No errors have been detected
300	SMC Linux Memory Map Failed
301	Insuffucient Buffer
302	SMC Not Ready
303	SMC No Free Buffer Segment
304	Error Abort
305	Error Pause
306	SMC Fpga Driver Not Initialized
310	License Serial Number Mismatch
311	License File Corrupted
312	License File Missing
313	License Tiling File Missing
314	License Tray File Missing
315	License Dll Automation File Missing
316	License Database File Missing
317	License Surface Marking File Missing
318	License Engraving File Missing

Appendix B - Error Codes

Value	Description
319	SMC Error in Syncmaster License
320	SMC Netlink Failed
321	SMC Netlink Send Message Failed
350	SMC Mailbox Size Exceeded
351	SMC Invalid Data Package
352	SMC System In Error State
353	SMC Using Backup Configuration
400	SMC Abort Detected
401	SMC No Buffer Available
500	SMC Stage Buffer Not Available
501	SMC Stage Not Initialized
502	SMC Stage Sync Update Fail
503	SMC Stage Not Enabled Configuration
504	SMC Stage Serial Communication Failed
505	SMC Stage Data Copy Error
506	SMC Stage Memory Creation Failed
507	SMC Stage Connection Timeout
508	SMC Stage Homing Failed
509	SMC Stage Tracking Not Enabled
510	SMC Stage Invalid Reply
706	Job Sequence Not Found

Value	Description
713	Xml Load Error
1001	Bad correction table data in file
1002	Inappropriate scale factor in correction table file
1003	General parsing error of correction table file
1100	Unidentified exception was caught
1101	Correction table memory allocation failed
1102	Error reading the Admin config file
1103	Error reading the Control config file
1104	Error reading the Laser config file
1105	Error reading the Lens config file
1106	Error reading the User config file
1107	Error reading the Performance config file
1108	Error reading the Servo config file
1109	Error reading the Vector Parameters config file
1110	Error reading the Sync Master config file
1120	Default config files being used because of a prior error detected
1200	Selected config file was not found
1201	Overwrite config file with backup
1300	Invalid XML node detected while parsing
1301	Invalid XML value detected while parsing
1310	Error parsing the Control config file

Value	Description
1311	Error parsing the Laser config file
1312	Error parsing the Lens config file
1313	Error parsing the User config file
1314	Error parsing the X axis data in the Servo config file
1315	Error parsing the Y axis data in the Servo config file
1316	Error parsing the Z axis data in the Servo config file
1317	Error parsing the ScanPack vector params
1318	Error parsing the ScanPack spiral params
1319	Error parsing the ScanPack circle params
1320	Error parsing the ScanPack point params
1321	Error parsing the ScanPack laser params

10 INDEX

- .
- .NET API format, 13
- .NET C#, 16
- 1**
 - 1 command. *See* Abort command
 - 10 command. *See* GetKFactor command
 - 14 command. *See* SetPerformanceGlobals command
 - 15 command. *See* ResetPerformanceGlobals command
 - 16 command. *See* OpenCOMPort command
 - 17 command. *See* CloseCOMPort command
 - 18 command. *See* COMWriteLine command
- 2**
 - 2 command. *See* TakeHostControl command
 - 200 command. *See* ClearJobList command
 - 203 command. *See* GetFlashJobFileList command
 - 204 command. *See* GetUSBJobFileList command, *See* GetUSBJobFileList command, *See* GetUSBJobFileList command
 - 205 command. *See* LoadFlashJob command
 - 206 command. *See* LoadUSBJob command
 - 207 command. *See* ExecuteJobOnce command
 - 208 command. *See* ExecuteJobContinuous command
 - 209 command. *See* GetJobStatus command
 - 21 command. *See* MotfCalFactor command
 - 211 command. *See* GetJobState command
 - 212 command. *See* GetJobElapsedTime command, *See* GetJobElapsedTime command, *See* GetJobElapsedTime command, *See* GetJobElapsedTime command
- 27 command. *See* GetZKFactor command
- 28 command. *See* GetYKFactor command
- 29 command. *See* GetControllerTemp command, *See* GetControllerTemp command, *See* GetControllerTemp command, *See* GetControllerTemp command
- 3**
 - 3 command. *See* ReleaseHostControl command
 - 35 command. *See* GetDigitalPort command
- 4**
 - 4 command. *See* GetHostControlStatus command
- 5**
 - 5 command. *See* GetHostInControl command
 - 500 command. *See* SetAdminPIN command
 - 501 command. *See* GetAdminPIN command
 - 502 command. *See* SetDHCPMode command
 - 503 command. *See* GetDHCPMode command
 - 504 command. *See* SetLocalGateway command
 - 505 command. *See* GetLocalGateway command
 - 506 command. *See* SetLocalIP command
 - 507 command. *See* GetLocalIP command
 - 508 command. *See* SetNodeFriendlyName command
 - 509 command. *See* GetNodeFriendlyName command
 - 510 command. *See* SetSubnetMask command
 - 511 command. *See* GetSubnetMask command
 - 512 command. *See* SetUserPIN command
 - 513 command. *See* GetUserPIN command

514 command. *See* SetCOMPortSpeed command
 515 command. *See* GetCOMPortSpeed command
 516 command. *See* SetCOMPortAssignments command
 517 command. *See* GetCOMPortAssignments command

6

6 command. *See* EnableBroadcasting command

7

7 command. *See* LoadHardwareDefaults command

8

8 command. *See* HardwareReset command

9

9 command. *See* GetRemoteIP command

A

Abort command, 238
 ActiveCorrectionTable parameter, 141
 ActuatorMin XML Tag, 78
 ActuatorUnits XML Tag, 72
 Address XML Tag, 46
 Admin XML Tag, 49
 AliveChannel XML Tag, 45
 An XML Tag, 75, 76
 Aperture XML Tag, 67
 API formats, 13
 APIPort XML Tag, 47
 APIPortSpeed XML Tag, 47
 ApplicationEvent command, 137
 ArcAbs command, 108
 AVer XML Tag, 29
 AxisDACConfig parameter, 147

B

BeginJob command, 137
 Bits XML Tag, 59
 BreakOK XML Tag, 47

BroadcastChannel XML Tag, 46

C

CalFlag XML Tag, 66
 CalibrateRectangularField XML Tag, 71
 ClearJobList command, 252
 Client XML Tag, 47
 CloseCOMPort command, 246
 CmdRangeCheckMode parameter, 93
 CmdRangeCheckMode XML Tag, 52
 CO₂ Laser Timing. *See* Laser Timing Control
 COMWriteLine command, 247
 Configuration XML Tag, 71, 72
 ConnectIP XML Tag, 30
 ConnectJob XML Tag, 30
 ContrlTemp XML Tag, 33
 ControlFile XML Tag, 44
 Coordinate System Conventions, 269
 Correction Table Support, 281
 CorrFile1 XML Tag, 51
 CorrFile2 XML Tag, 51
 CorrFile3 XML Tag, 51
 CorrFile4 XML Tag, 51
 CurrentDIO XML Tag, 34

D

Data XML Tag, 28, 30, 32, 34, 44, 51, 57, 58, 66, 69, 70, 81, 82, 83, 84, 85, 86, 87
 DataChannel XML Tag, 44
 DebugPort XML Tag, 48
 DebugPortSpeed XML Tag, 48
 Delays, 272
 DeleteAllSegments command, 196
 DeleteSegment command, 196
 Description XML Tag, 70
 DesignErrorComponents XML Tag, 73
 DFMPort XML Tag, 48
 DFMPortSpeed XML Tag, 48
 DigitalIOPolarity XML Tag, 56
 DisableSegment command, 197
 Distortion
 Composite Image Field Distortion, 281
 Pillow Distortion, 280
 Pincushion Distortion, 280

X-Y Mirror Induced Distortion, 279
 DistortionFactor XML Tag, 73
 Duty XML Tag, 59
 Dx XML Tag, 77
 Dy XML Tag, 77
 Dz XML Tag, 77

E

e1e2Coeffs XML Tag, 76
 E1E2Spacing XML Tag, 75
 EnableBroadcasting command, 240
 EnableParkPosition command, 108
 EnableSegment command, 197
 EnableStreamToFile XML Tag, 44
 EndJob command, 138
 EventChannel XML Tag, 45
 ExecuteJobContinuous command, 255
 ExecuteJobOnce command, 254
 ExtPauseControl XML Tag, 56
 ExtPwrCtrl XML Tag, 59

F

Fiber Laser Timing. *See* Laser Timing Control
 FieldOffset parameter, 142
 FieldOrientation parameter, 142
 Fire-on-the-fly, **Error! Not a valid bookmark in entry on page 152**
 FixedFreq XML Tag, 58
 FixedPW XML Tag, 58
 FixedWatts XML Tag, 58
 FocalLen XML Tag, 67
 FPGAfirmVer XML Tag, 29
 FreePermStorage XML Tag, 29
 FreeTempStorage XML Tag, 29
 FreeUSBStorage XML Tag, 29
 Freq XML Tag, 59
 FriendlyName XML Tag, 30, 46

G

GalvoCmdDelayComp command, 116
 GalvoCmdMarker command, 138
 GetAdminPIN command, 260
 GetCOMPortAssignments command, 268
 GetCOMPortSpeed command, 266

GetControllerTemp command, 249
 GetDHCPMode command, 261
 GetErrorCodeDescription method, 235
 GetFlashJobFileList command, 252
 GetHostControlStatus command, 239
 GetHostInControl command, 240
 GetJobElapsedTime command, 256, 257, 258
 GetJobState command, 256
 GetJobStatus command, 255
 GetKFactor command, 242
 GetLocalGateway command, 262
 GetLocalIP command, 263
 GetNodeFriendlyName command, 263
 getPriorityData method, 232
 GetRemoteIP command, 242
 GetSubnetMask command, 264
 GetUSBJobFileList command, 252, 258, 259
 GetUserPIN command, 265
 GetYKFactor command, 248
 GetZKFactor command, 248

H

HardwareReset command, 241
 HeadParameters XML Tag, 73, 76
 HeadSerialNumber XML Tag, 46
 HeadType XML Tag, 71
 HSN XML Tag, 30

I

InitPosition XML Tag, 55
 InsGenMode XML Tag, 55
 Installation location, 14
 Interlock XML Tag, 33, 59
 IntlockConfig XML Tag, 53
 IP XML Tag, 30
 IPAddress XML Tag, 49
 IPGateway XML Tag, 49
 IPMode XML Tag, 49
 IPRetries XML Tag, 49
 IPSubnet XML Tag, 49
 IPTimeout XML Tag, 49
 IPTryagain XML Tag, 49
 ISRGenMode XML Tag, 55

J

Job parameters, **Error! Not a valid bookmark in entry on page 91**

- JobDataCntr command, 138
- JobDataCntr XML Tag, 34
- JobMarker command, 139
- JobMarker XML Tag, 34
- JobTimer command, 140
- JumpAbs command, 109
- JumpAbsEx command, 110
- JumpAbsList command, 111
- JumpAndFireList command, 114, 117
- JumpDelay parameter, 93
- JumpRelEx command, 113
- JumpSpeed parameter, 94
- JumpStepTime parameter, 95

L

Laser Marking Terms and Definitions, 271

Laser Timing Control, **Error! Not a valid bookmark in entry on page 282**

- CO₂ Laser Timing, 289
- Fiber Laser Timing, 301
- Laser Timing Emulation, 288
- Nd05C3:YAG Emulation Mode-3 Timing, 295
- Nd05C3YAG Emulation Mode-1 Timing, 291
- Nd05C3YAG Emulation Mode-2 Timing, 293
- Nd05C3YAG Emulation Mode-4 Timing, 297
- Nd05C3YAG Emulation Mode-5 Timing, 299

Laser Timing Emulation. See Laser Timing Control

- LaserEnable command, 131
- LaserEnableDelay parameter, 125, 146
- LaserEnableDelay XML Tag, 63
- LaserEnableTimeout parameter, 125
- LaserEnableTimeout XML Tag, 63
- LaserFile XML Tag, 51
- LaserFire command, 132
- LaserFPK XML Tag, 63
- LaserModDelay parameter, 126
- LaserModDelay XML Tag, 63
- LaserModeConfig parameter, 147
- LaserModeConfig XML Tag, 60

- LaserModType parameter, 130
- LaserOffDelay parameter, 126
- LaserOn command, 132
- LaserOnDelay parameter, 127
- LaserPipelineDelay parameter, 128
- LaserPipelineDelay XML Tag, 52
- LaserPort XML Tag, 48
- LaserPortSpeed XML Tag, 48
- LaserPower XML Tag, 85, 86
- LaserPowerDelay parameter, 128
- LaserPowerDelay XML Tag, 64
- LaserPulse parameter, 129
- LaserRegulation command, 166
- LaserScribe command, 166
- LaserSignalOff command, 133
- LaserSignalOn command, 133
- LaserStandby parameter, 127
- LaserStandby XML Tag, 63
- LaserTiming parameter, 130
- LastError Code Descriptions, 308
- LastError XML Tag, 29
- Layer XML Tag, 80, 81
- Lens XML Tag, 73
- LensFile XML Tag, 51
- LensFocalLength-mm XML Tag, 74
- LensMaxMechHalfAngle-deg XML Tag, 75
- LensName XML Tag, 66
- LissajousWobble params, 101
- LoadFlashJob command, 253
- LoadHardwareDefaults command, 241
- LoadUSBJob command, 253
- LocalMode XML Tag, 46
- LoggingLevel XML Tag, 49
- LongDelay command, 140
- LsrName XML Tag, 58
- LsrType XML Tag, 58

M

- MAC XML Tag, 30
- MarkAbs command, 119
- MarkAbsEx command, 120
- MarkAbsList command, 121
- MarkDelay parameter, 95

MarkRel command, 123
 MarkRelEx command, 124
 Marks and Jumps, 270
 MarkSpeed parameter, 96
 MarkSpeed XML Tag, 85, 86
 MarkStepTime parameter, 97
 MicroStepMode XML Tag, 55
 Micro-vectoring, 272
 Mirrors XML Tag, 73
 mmToActuatorSpaceTransform XML Tag, 76, 78
 MotfCalFactor parameter, 160
 MotfCalFactor XML Tag, 52
 MotfCalGain XML Tag, 51
 MotfCapable XML Tag, 51
 MotfDelayComp parameter, 161
 MotfDirection parameter, 161
 MotfDirection XML Tag, 52
 MotfEnable command, 164
 MotfMode parameter, 162
 MotfMode XML Tag, 52
 MotfResetJump command, 165
 MotfTriggerEvent parameter, 163
 MotfTriggerEx parameter, 162
 MotfWaitForTrigger command, 165
 MotionPort XML Tag, 47
 MotionPortSpeed XML Tag, 48
 MSN XML Tag, 28

N

Nd05C3YAG Emulation Mode-1 Timing. *See* Laser Timing Control
 Nd05C3YAG Emulation Mode-2 Timing. *See* Laser Timing Control
 Nd05C3YAG Emulation Mode-3 Timing. *See* Laser Timing Control
 Nd05C3YAG Emulation Mode-4 Timing. *See* Laser Timing Control
 Nd05C3YAG Emulation Mode-5 Timing. *See* Laser Timing Control
 NetAssign XML Tag, 30
 NetMask XML Tag, 30

O

ObjExtVer XML Tag, 29
 Offset parameter, 143, 144
 OnDataEvent method, 221
 OnMessageEvent Message Types, 215
 OpenCOMPort command, 245

P

Pendant XML Tag, 47
 PendantPort XML Tag, 47
 PendantPortSpeed XML Tag, 47
 PerformanceFile XML Tag, 51
 Period XML Tag, 85, 86
 PermStoragePath XML Tag, 29
 PincushionFactor XML Tag, 73
 PixelMap command, 158
 PolyDelay parameter, 97
 Port XML Tag, 30, 44, 45, 46
 Predefined Application Message Event Codes, 216
 PreserveCalFactors XML Tag, 71
 PriorityChannel XML Tag, 45
 Pulse XML Tag, 58
 PulseWidth XML Tag, 85, 86
 PVer XML Tag, 29

R

RasterLine command, 159
 RasterMode command, 158
 RasterParams command, 158
 ReferenceInformation XML Tag, 70, 71
 RefSurfaceToWorkSurfaceDist-mm XML Tag, 74
 ReleaseHostControl command, 239
 ResetPerformanceGlobals command, 244
 Retransmit XML Tag, 46
 Revision history, 5
 Rotation XML Tag, 83
 RTCCCompatibility parameter, 145
 RTCCCompatibility XML Tag, 55
 RunSegment command, 195

S

ScanScript embedded scripting language, 13
 Segment command, 194
 sendJobData method, 210
 sendPriorityData method, 222
 sendStreamData method (overload 1), 184, 186, 188, 190, 191
 Sequence command, 195
 ServoConfig parameter, 150
 Set command, 140
 SetAdminPIN command, 259
 SetCOMPortAssignments command, 267
 SetCOMPortSpeed command, 266
 SetDHCPMode command, 260
 SetLocalGateway command, 261
 SetLocalIP command, 262
 SetMotfEncoderRate command, 248
 SetNodeFriendlyName command, 263
 SetSubnetMask command, 264
 SettleCheckMode parameter, 180, 182
 SetUserPIN command, 265
 SMC Hardware Reference Manual, 5
 SourceLensID XML Tag, 70
 SourceScanHeadID XML Tag, 70
 SourceSpacerID XML Tag, 70
 StartupJob XML Tag, 56
 StateCode XML Tag, 29
 StreamFile XML Tag, 45
 SupplementalLayers XML Tag, 80, 81

T

TableCreationDate XML Tag, 71
 TableDataHasBeenCorrectedFromDesign XML Tag, 72
 TableParams XML Tag, 70, 79
 TableRevision XML Tag, 70
 TableStructure XML Tag, 78, 79
 TakeHostControl command, 238
 Tbl1Rotation XML Tag, 68
 Tbl1XGain XML Tag, 68
 Tbl1XOff XML Tag, 68
 Tbl1YGain XML Tag, 68
 Tbl1YOff XML Tag, 68

Tbl2Rotation XML Tag, 68
 Tbl2XGain XML Tag, 68
 Tbl2XOff XML Tag, 68
 Tbl2YGain XML Tag, 68
 Tbl2YOff XML Tag, 68
 Tbl3Rotation XML Tag, 69
 Tbl3XGain XML Tag, 68
 Tbl3XOff XML Tag, 68
 Tbl3YGain XML Tag, 69
 Tbl3YOff XML Tag, 68
 Tbl4Rotation XML Tag, 69
 Tbl4XGain XML Tag, 69
 Tbl4XOff XML Tag, 69
 Tbl4YGain XML Tag, 69
 Tbl4YOff XML Tag, 69
 ThirdAxisPresent XML Tag, 71
 Transform parameter, 144
 TransformEnable parameter, 145

U

Units parameter, 91
 UseExtPwrCtrl XML Tag, 59
 User XML Tag, 48
 UserFile XML Tag, 51
 UserVar1 XML Tag, 82
 UserVar2 XML Tag, 82
 UserVar3 XML Tag, 82
 UserVar4 XML Tag, 82
 UserVar5 XML Tag, 82
 UserVar6 XML Tag, 83
 UsingFile command, 197

V

VariJumpDelay parameter, 97
 VariPolyDelayFlag parameter, 98
 VelocityComp command, 178
 VisPtr XML Tag, 59
 Volts XML Tag, 59

W

WaitForIO command, 134
 Watts XML Tag, 59
 WattsUnits XML Tag, 58
 Win32 DLL API format, 13

Wobble mode, 99
 Wobble parameter, 99
 Wobble table, 100
 WobbleEnable command, 102
 WriteAnalog command, 135
 WriteDigital command, 135

X

XActPos XML Tag, 32
 XActuatorStride XML Tag, 78
 x-axis XML Tag, 80, 81
 XGain XML Tag, 83
 XGalvoMechHalfAngle-deg XML Tag, 74
 XMirrorToObjectiveDist-mm XML Tag, 75
 XML in the API, 20
 XML Tags
 Administration Configuration, 44
 Broadcasted Status Information, 32
 Broadcasted System Information, 28
 Controller Configuration, 51
 Correction Table, 70, 80, 82, 83
 Laser Configuration, 58, 60
 Lens Configuration, 66, 68
 Performance Adjustments Table, 84, 85, 86
 X-NumCols XML Tag, 78
 XOff XML Tag, 83
 XOffset XML Tag, 85, 86
 XPos XML Tag, 32
 XPosAck XML Tag, 32
 XPower XML Tag, 33
 XStatus XML Tag, 33
 XTemp XML Tag, 32
 XtoYMirrorDist-mm XML Tag, 74
 Xx XML Tag, 76
 Xy XML Tag, 77
 XY2AddressingMode XML Tag, 54
 XY2AxisDisable parameter, 105, 106, 107
 XY2ErrorCheckMode parameter, 103, 104

XY2FrameRate XML Tag, 54
 XY2StatusTiming XML Tag, 54
 XYCalFactor parameter, 92
 Xz XML Tag, 77

Y

YActPos XML Tag, 32
 YActuatorMin XML Tag, 78
 YActuatorStride XML Tag, 78
 y-axis XML Tag, 80, 81
 YGain XML Tag, 83
 YGalvoMechHalfAngle-deg XML Tag, 74
 YMirrorToRefSurfaceDist-mm XML Tag, 74
 Y-NumRows XML Tag, 79
 YOff XML Tag, 83
 YOffset XML Tag, 85, 87
 YPos XML Tag, 32
 YPosAck XML Tag, 32
 YPower XML Tag, 33
 YStatus XML Tag, 33
 YTemp XML Tag, 33
 Yx XML Tag, 76
 Yy XML Tag, 77
 Yz XML Tag, 77

Z

ZActuatorMin XML Tag, 79
 ZActuatorStride XML Tag, 79
 z-axis XML Tag, 80, 81
 ZCalFactor parameter, 92
 ZCalFactorCoeffs XML Tag, 75
 ZMode XML Tag, 66
 Z-NumLayers XML Tag, 79
 ZOffset XML Tag, 85, 87
 Zx XML Tag, 77
 Zy XML Tag, 77
 Zz XML Tag, 77

This page is left blank intentionally



Novanta

PHOTONICS

Engineered by Cambridge Technology, part of Novanta

Novanta Headquarters, Bedford, USA

Phone: +1-781-266-5700

Email: photonics@novanta.com

Website: www.novantaphotonics.com

1040-0012 Revision A

February 2022

© 2022, Novanta Corporation. All rights reserved.