AEROTECH

IFOV Setup Guide for Automation1 Studio

USER GUIDE

Revision 1.00

GLOBAL TECHNICAL SUPPORT

Go to the Global Technical Support Portal for information and support about your Aerotech, Inc. products. The website supplies software, product manuals, Help files, training schedules, and PC-to-PC remote technical support. If necessary, you can complete Product Return (RMA) forms and get information about repairs and spare or replacement parts. To get help immediately, contact a service office or your sales representative. Include your customer order number in your email or have it available before you call.

List of Figures

List of Equations

Table of Contents

Safety Procedures and Warnings

IMPORTANT: For warning and safety information and other important information about your drives and laser scan head, refer to the hardware manuals. To download your manuals from www.aerotech.com, go to the **Automation1** (drives) and **Mechanical** (laser scan head) sections of Manuals & Help Files.

DANGER: Refer to your laser manual and obey all of the safety precautions for the laser. To prevent injury and equipment damage, make sure that the laser output signals are configured correctly before you do an operation that causes the laser to fire.

DANGER: The danger to your eyes increases when optical instruments are used in conjunction with the scan head:

- Wear certified laser safety eye protection.
- Do not stare into the laser beam, put your body parts in the laser area, or expose yourself to reflections from powerful beams.

Chapter 1: Introduction to Infinite Field of View (IFOV)

Infinite Field of View (IFOV), by Aerotech, is a feature of the Automation1 PC-based motion controller that synchronizes one set of orthogonal servo axes with a 2-axis galvo scanner. This system combines the capabilities of a 2-axis galvo scanner, which has high dynamic galvanometer motors, with the extended travel range of linear servo stages. As a result, you can process parts that are much larger than the optical field of view (FOV) of a galvo scanner.

The Aerotech IFOV implementation controls the servo and galvo scanner motion from a single trajectory, which divides motion between the servo and galvo scanner axes in real time. It also minimizes optical distortion typical of F-Theta optics by optimizing the trajectory to keep the laser spot biased to the center of the FOV during all motion. As a result, IFOV makes significant throughput improvements and removes stitching errors typical to step-and-scan operations.

In addition, IFOV is a feedback-based control algorithm that uses encoder signals from the servo and galvo scanners. This system dynamically corrects for position errors from the servo stages. The result is an optimized motion trajectory where the laser spot moves at a defined coordinated speed target. The laser is triggered based on its actual position, which is derived from real-time feedback of the galvo scanner and servo axes.

1.1. IFOV Operating Principles

IFOV is a motion controller feature that requires a specific hardware and software configuration. When IFOV is enabled, you can program your 2D galvo scanner pattern based on the travel limits of the servo stages instead of the optical field of view (FOV) limits of the galvo scanner. IFOV operates on two important functions:

- The servo stages will move to let the galvo scanner operate within its FOV on any trajectory that is accessible while the FOV translates across the travel range of the servo stage.
- The system prioritizes stage motion commands to keep the laser spot as near to the galvo scanner FOV center as possible. This minimizes spot size distortion caused by F-Theta lens aberrations.

The position command and position feedback data of the galvo scanner axis will match the correct trajectory path on the workpiece. But the positions commanded to the servo axis are generated by the IFOV algorithm to execute the two important functions previously mentioned. For this reason, the servo axis feedback, when plotted as data, will be unique to each programmatic change to velocity and other parameters discussed throughout this guide. Thus, you must consider that you are programming the galvo scanner axes to move the beam with its defined spot size along its intended path at a specified coordinated velocity and acceleration. It is also important to think about what the galvo scanner and servo axes do to make this possible.

To better understand this concept, refer to Figure 1-1. It shows the different components of the IFOV system (servo stage travel, galvo scanner field of view (FOV), and user part profile) and how they operate while IFOV is enabled. Figure 1-1 outlines the total travel area of the servo stage and the galvo scanner FOV that moves within it. The black line indicates the path of the servo stages and the resulting features marked by the galvo scanner.

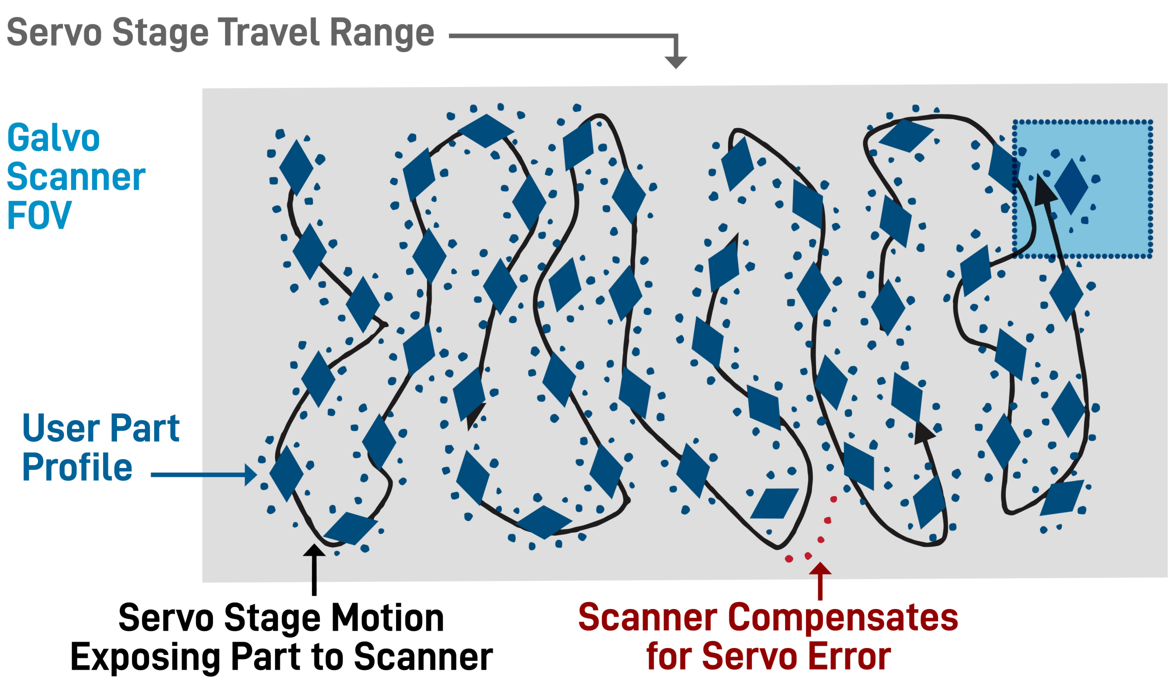

Servo Stage Travel Range
Galvo Scanner FOV
User Part Profile
Servo Stage Motion Exposing Part to Scanner
Scanner Compensates for Servo Error

Figure 1-1: IFOV System Components in Operation

For a visual representation of the IFOV system components in operation, see Figure: IFOV System Components in Operation (animation).

1.2. IFOV Machine Configurations

In Figure 1-2 that follows, the IFOV configuration shows you how an Automation1 PC, galvo scanner drives, servo drives, and mechanics are connected and driven by the Automation1 Studio software.

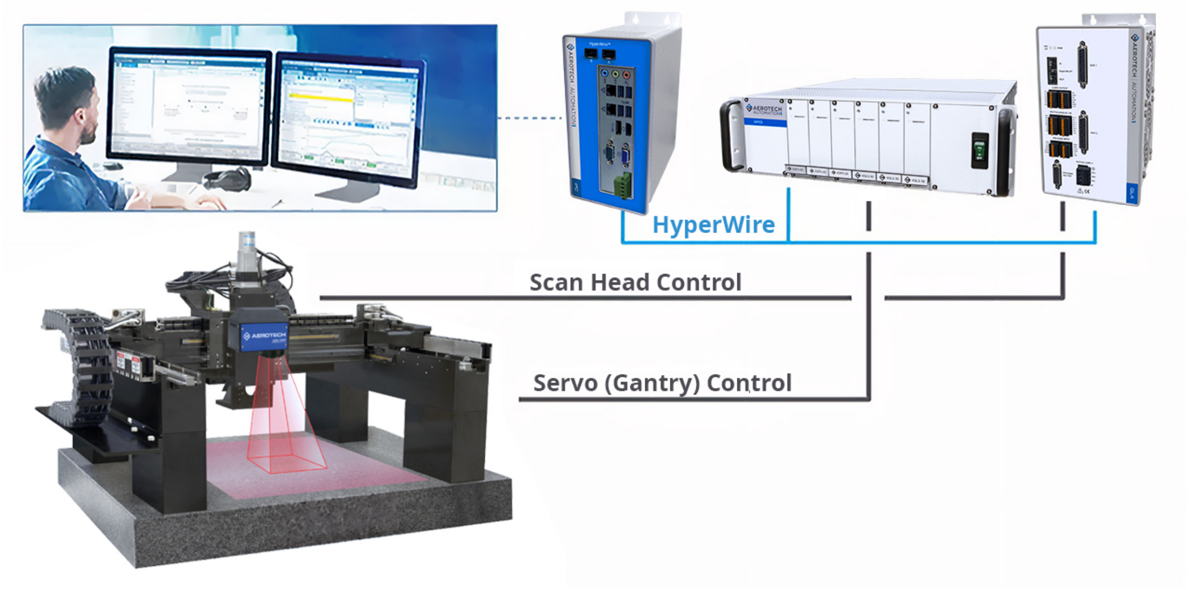


Figure 1-2: IFOV Machine Configuration

A correct machine configuration, as shown in Figure 1-2, for an Infinite Field of View (IFOV) system is defined as one of the following:

- One two-axis galvo scanner drive (Automation1 GL4) driving a single AGV galvo scanner, which has two galvo scanner motors.
- One galvo scanner interface (Automation1 GI4) driving a third-party 2-axis galvo scanner through either an (18-bit) XY2-100 or (24-bit) XY3-100 protocol.

Both options are powered by an Automation1 PC-based motion controller and two axes of linear or rotary motion. Some typical machine configurations that match this definition include:

1.2.1. Configuration A: Moving galvo scanner on a gantry

This IFOV configuration shows an AGV galvo scanner mounted to a gantry motion system. The laser galvo scanner moves over a work area.

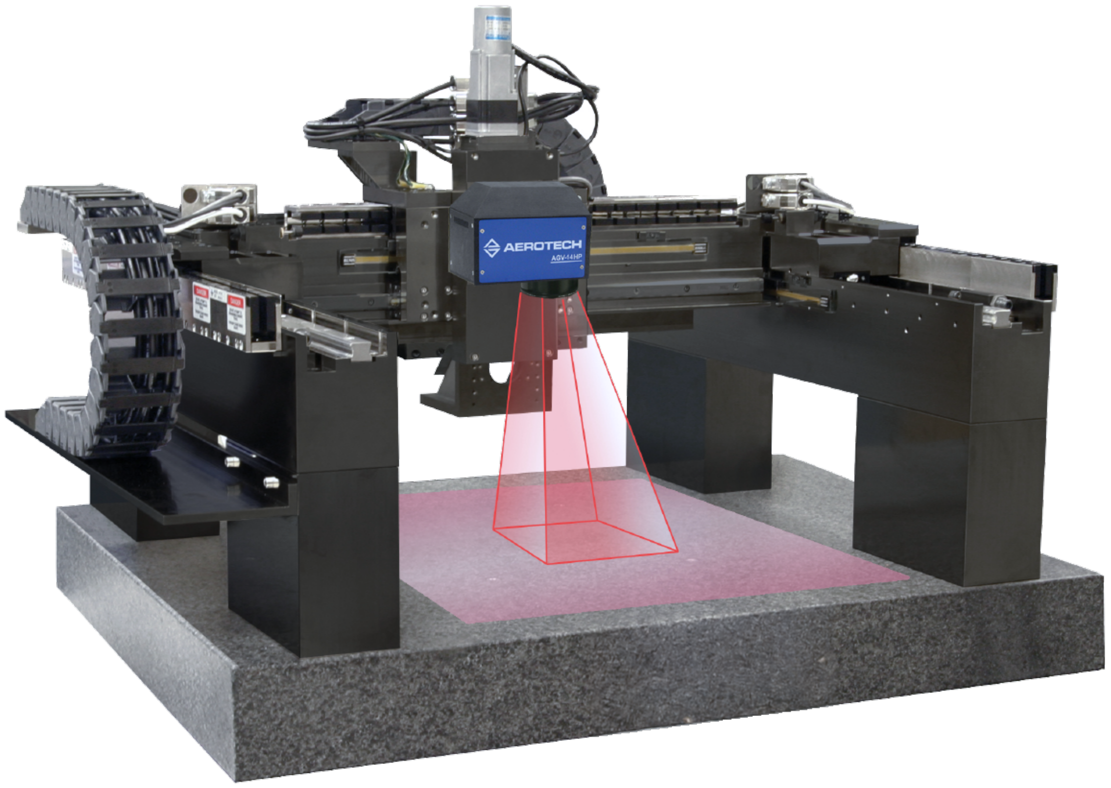


Figure 1-3: AGV Galvo Scanner Mounted to Gantry

1.2.2. Configuration B: Static galvo scanner over an XY motion table

This IFOV configuration shows an XY linear stage stack with a static galvo scanner.

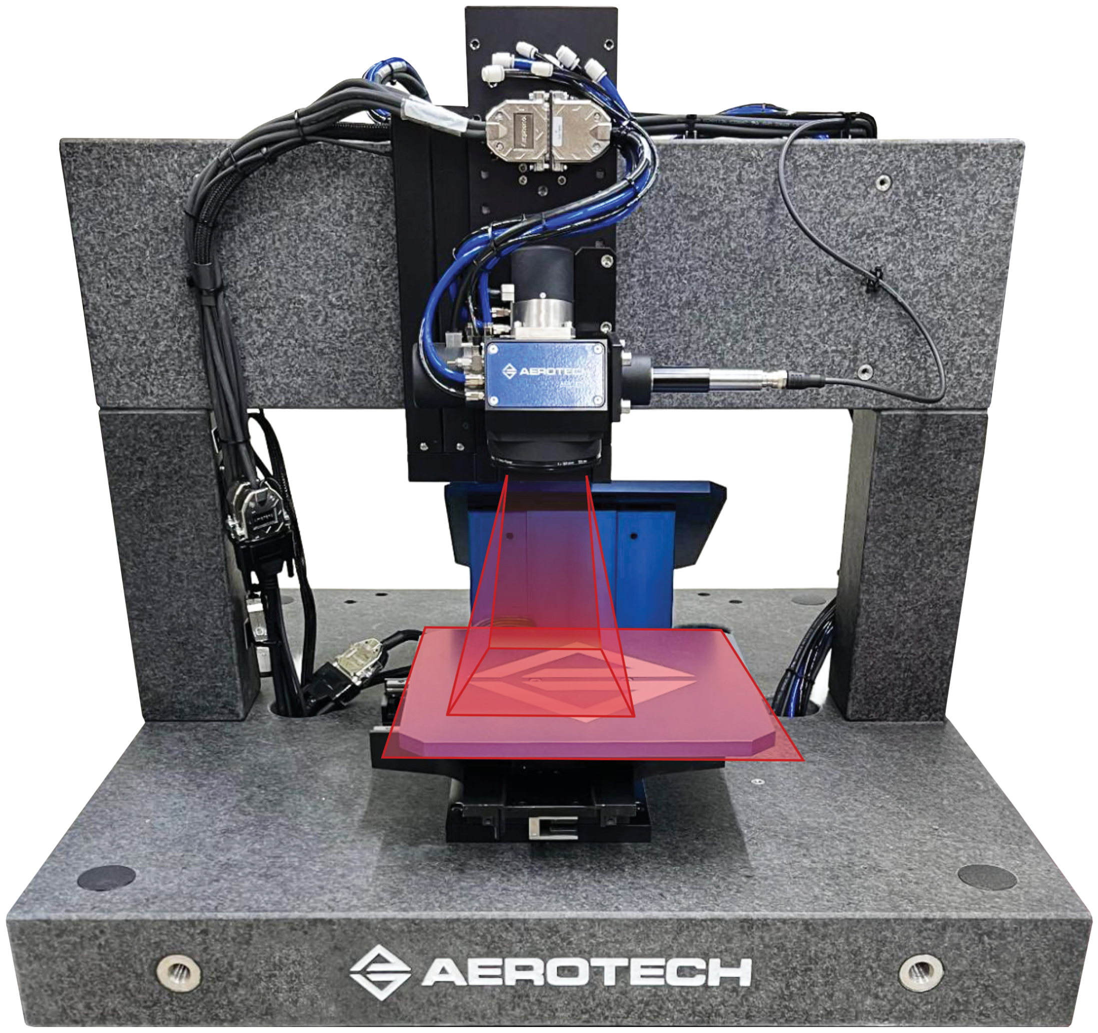


Figure 1-4: XY Linear Stage Stack - Static Galvo Scanner

1.2.3. Configuration C: Split XY configuration - galvo scanner mounted to a linear axis

This IFOV configuration shows a split XY motion system with a galvo scanner on a single cross-axis mounted to an orthogonal linear axis.

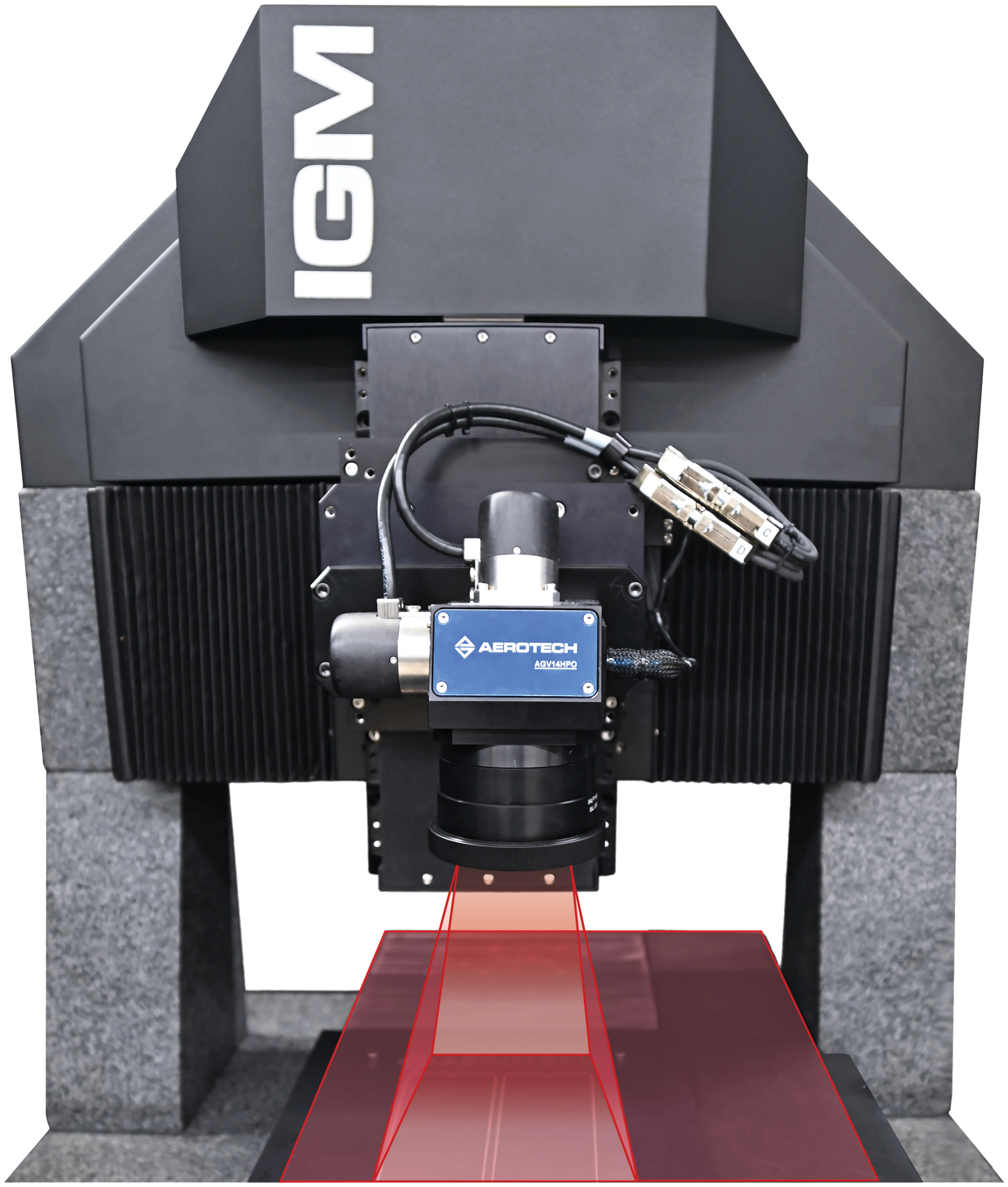


Figure 1-5: Split XY Configuration - Moving Galvo Scanner

Other machine configurations might be available for IFOV. Aerotech can suggest or design more configurations that are not discussed in this guide. If your IFOV configuration is not discussed here or you have questions about how to correlate your unique setup to this guide, use the Global Technical Support Portal to contact Aerotech Global Technical Support.

1.3. IFOV Guide Demonstration System

To better show the important operating principles and concepts of IFOV, a demonstration system is discussed and used throughout this guide. It uses Configuration C. As you move through the guide, refer back to this section for the axis names and directions of travel shown in Figure 1-6. The system in this figure has two linear servo axes that this guide refers to as **SX** and **SY**, and two galvo scanner axes that this guide refers to as **GX** and **GY**. The motion that is executed between the galvo scanner axes and linear servo axes occurs between collinear axis pairs. **SX** and **GX** are a collinear pair. **SY** and **GY** are a collinear pair.

The orientation of the machine coordinate system defines the positive direction of travel for all axes.

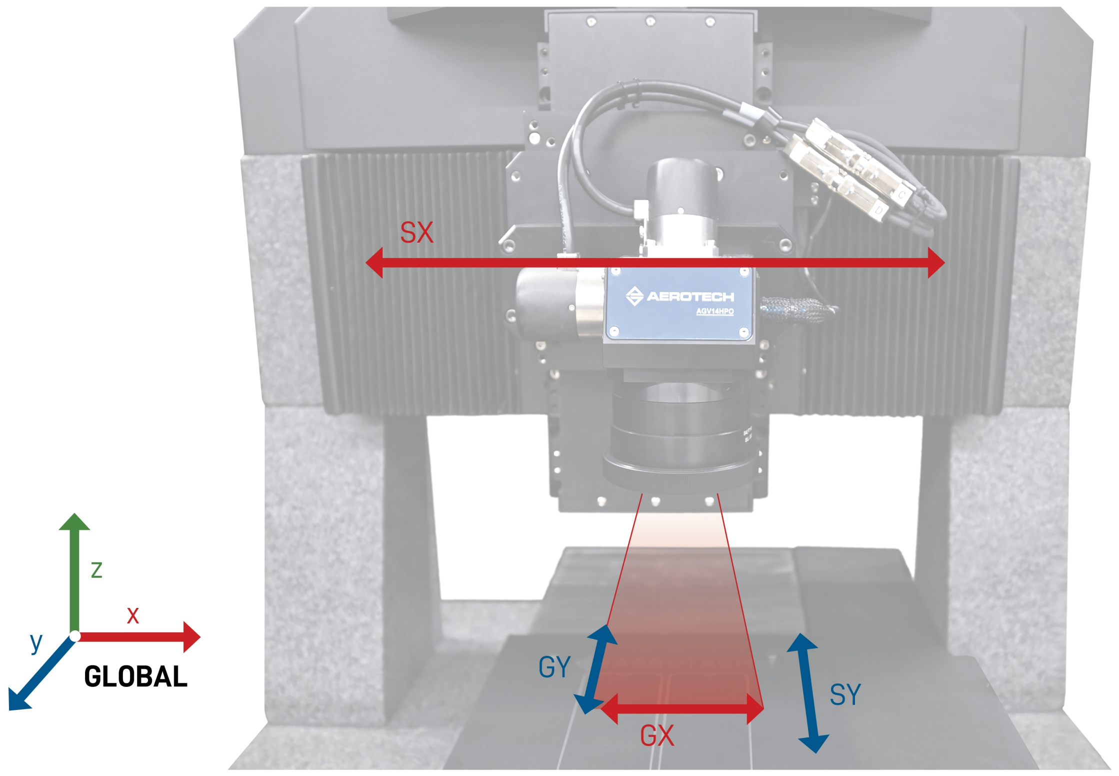


Figure 1-6: Split XY Configuration with Moving Galvo Demonstration System for the IFOV Guide

Chapter 2: IFOV Setup and Configuration Overview

This guide will help you make sure the IFOV system is configured correctly. For optimal system performance, do the steps in this guide in the order that they occur. The full IFOV setup is divided into the sections that follow:

1. **Hardware Selection and Setup**
   - Select the correct controller, servo drive, stage, galvo scanner, and cable hardware.
   - Use applicable wiring connections for axis feedback and drive encoder signals.
   - For more information, refer to Chapter 3: Hardware Selection and Setup.
2. **System Software Configuration and Setup**
   - Configure the Automation1 Machine Controller Definition (MCD) file. See Machine Controller Definition Files for more information.
   - Configure axis motion direction and feedback scaling.
   - Configure galvo scanner axis feedback.
   - Configure servo encoder signal echoing to galvo scanner axes.
   - Configure IFOV with Position Synchronized Output (PSO).
   - For more information, refer to Chapter 4: IFOV Software Configuration.
3. **IFOV System Verification**
   - Verify that the galvo scanner axes are tracking the servo axes.
   - Verify the polarity of auxiliary encoder cable signals.
   - For more information, refer to Chapter 5: IFOV System Verification.
4. **IFOV System Calibration**
   - Verify the feedback scaling for each galvo scanner axis.
   - Galvo scanner 2D calibration. See Calibration Module for more information.
   - Verify servo to galvo scanner axis orthogonality.
   - For more information, refer to Chapter 6: IFOV System Calibration.
5. **Write Your IFOV Program**
   - Commands you can use to configure IFOV. See IFOV Functions for more information.
   - A ready-to-use AeroScript Library that helps expedite how you use and configure IFOV.
   - Commands that turn IFOV on and off.
   - For more information, refer to Chapter 8: Write Your IFOV Program.

Chapter 3: Hardware Selection and Setup

The first step to correctly configure an IFOV system is to make sure you select the correct equipment, which includes servo stages and galvo scanner mechanics, drives or interfaces, cables, PC, and software.

3.1. The Automation1 PC-Based Controller

HARDWARE: IFOV is not compatible with Aerotech drive-based controllers, referred to as i-drives (e.g., iXR3, iXC4, etc.). These products are limited to 20 KHz motion. They do not support laser scan devices, such as AGV products or the Automation1-GI4 laser scan-head interface devices.

The IFOV feature is compatible only with the PC-based motion controller product. You must run the Automation1-iSMC and Automation1-MDK installers on the PC in order to use IFOV. You can purchase an Automation1 iPC from Aerotech that is preconfigured with iSMC and MDK. You can also purchase a PC from the PC Selection, Configuration, and Optimization for Aerotech Controllers guide, and set up the iSMC and MDK manually. See Install the Software for more information.

After you set up your PC correctly and it is running the Automation1-iSMC and MDK, you must make sure the correct hardware is connected and configured. Then you can use IFOV as a built-in feature.

3.2. Selecting Automation1 Servo and Galvo Scanner Drives

As previously discussed, two servo axes and two galvo scanner axes are required to set up IFOV.

For servo axis drives, you can use Automation1 XC/XL drives or an XR3 drive rack to control the two servo axes. The drives you select must have the correct encoder signal type to echo the encoder signals. They must also have the correct hardware multiplication (MX) configuration to enable the encoder signals to be output from the servo axis drive to the galvo scanner drive or interface device. This enables the galvo scanner to use the servo axis feedback for the IFOV motion and combine PSO signals between the different drives.

HARDWARE: Not all Automation1 XC/XL drives are compatible with IFOV systems because of the limitations in their auxiliary encoder output hardware:

- The -MX0 multiplier option is compatible with IFOV systems. But its encoder signal output, which is available from both SYNC and Auxiliary Encoder connectors, is limited to a square-wave quadrature signal without hardware multiplication. You might also need to increase the scale resolution of the specified linear axes to enable correct axis tracking.
- The -MX1 multiplier option does not support IFOV systems.

For optimal performance, Aerotech recommends that you use enhanced drives (e.g., XC2e, XC4e) with the -MX2 or -MX3 multiplier option. Their advanced hardware can multiply the encoder signals and divide them down to enable correct matching between the servo and galvo signals.

For galvo scanner drives, the Automation1-GL4 or Automation1-GI4 galvo scanner interface is the supported product you must use to set up IFOV in Automation1 Studio. The standard configuration for these products is sufficient. No special options or considerations are required.

3.3. Selecting a 2-Axis Galvo Scanner

Aerotech recommends its AGV 2-axis galvo scanner products, such as the AGV-HPO and AGV-XPO. This is because IFOV was designed to operate correctly with these products. Aerotech AGV 2-axis galvo scanners use dual-loop digital encoders powered by the high-performance servo loop contained within the

Automation1 GL4. When you use these products for your IFOV configuration, you will get the most accurate tracking for motion and Position Synchronized Output (PSO). You can also use Automation1 to collect trajectory feedback data.

3.3.1. Using the Automation1 GI4 with Third-Party Galvo Scanners

HARDWARE: The Automation1 GI4 is a laser scan-head interface device that converts trajectory information (usually sent to an Aerotech drive-and-stage combination) into XY2-100 or XY3-100 serial communication, which is not bi-directional. This means IFOV will command the scanner to move, but the controller cannot monitor its tracking errors. This will create limitations for IFOV that are specific to this hardware configuration (i.e., Automation1 GI4 with a third-party scanner). An important limitation to know about this configuration is that Position Synchronized Output (PSO) is not supported, but Part-Speed PSO is supported.

The Automation1 GI4 (3-axis galvo scanner interface) is an Automation1 drive interface device designed to be connected to a third-party galvo scanner that supports either XY2-100 or the higher resolution XY3-100 protocol. This device converts Automation1 trajectory commands into these serial protocols and lets you use a standard galvo scanner interface to connect most third-party galvo scanner devices. If you set up IFOV with this hardware, it follows the same process shown in this guide. But there are some special considerations for the system configuration that you must follow. These considerations will be discussed in their related sections, shown by **TIP** notes in those sections.

3.4. Selecting Linear Servo Motion Axes

Usually, if you select standard resolution encoder signals (-E1 encoder option on most PRO/ANT stages) for standard Aerotech stages (with 20-micrometer scale pitch) and a drive with -MX2, IFOV performance will be satisfactory.

If you use hardware that is different from this configuration, you must make sure that the servo axis encoder resolution is sufficient for accurate tracking. If your servo stages have low encoder resolution (e.g., you do not use the -MX2 option on the drives), the GL4 controller's ability to accurately track the servo axis position feedback will be compromised, which causes reduced IFOV tracking performance. For information about better hardware options, contact your Aerotech sales representative. See the Contact Sales section of www.aerotech.com.

3.5. Cables and Wiring

To connect the drives, galvo scanner, and stages necessary to make the IFOV system run correctly, use only Aerotech-supplied cables specified by the equipment manuals and cable guides. Refer to your specific product manuals for more information. Aerotech offers standard cables for all Aerotech drive-and-servo-stage combinations, cables to network two or more axes, and cables to send encoder channel signals from one axis to the next axis. You can download these manuals from the **Mechanical** or **Automation1** section of Manuals & Help Files at www.aerotech.com.

Use the HyperWire cables to connect the drive electronics to the HyperWire card of the computer. HyperWire connections and communications are serial, which means that each axis is within a daisy chain that starts from the PC and goes to the input of the first axis (HyperWire IN port), and then from the output of that first axis (HyperWire OUT port) to the input of the next drive. You must continue this sequence for each drive in the system. For more information about HyperWire, see HyperWire Communication and Connect the System.

Use the correct SYNC or Auxiliary output cables specified by Aerotech to connect the servo drives to the Automation1 GL4 or GI4. These cables enable the galvo scanner to see the feedback from each servo axis in real time. Each drive manual tells you which cables to use or specifies the cable connections necessary for custom cables to be manufactured. If you have questions about cables, refer to your drive manuals. You can also Contact Sales at www.aerotech.com.

If you purchased an integrated system from Aerotech, it includes interconnect cables and a system interconnect drawing. Refer to the system interconnect drawing supplied by Aerotech and use the interconnect cables to connect the motors, stages, and galvo scanners to the drives. If you need help, use the Global Technical Support Portal to contact Aerotech Global Technical Support.

3.5.1. Servo Axis Encoder Wiring

To configure IFOV correctly, you must manually wire the encoder of each servo axis to the corresponding galvo scanner axis to make sure the software setup is correct for these encoder signals. Thus, make sure they are wired correctly before you try to configure the software. Correct wiring requires you to make either a SYNC connection or an Auxiliary Encoder connection between the collinear axis drive pair. Figure 3-1 shows the IFOV hardware setup, which uses either SYNC cables or Auxiliary Encoder cables.

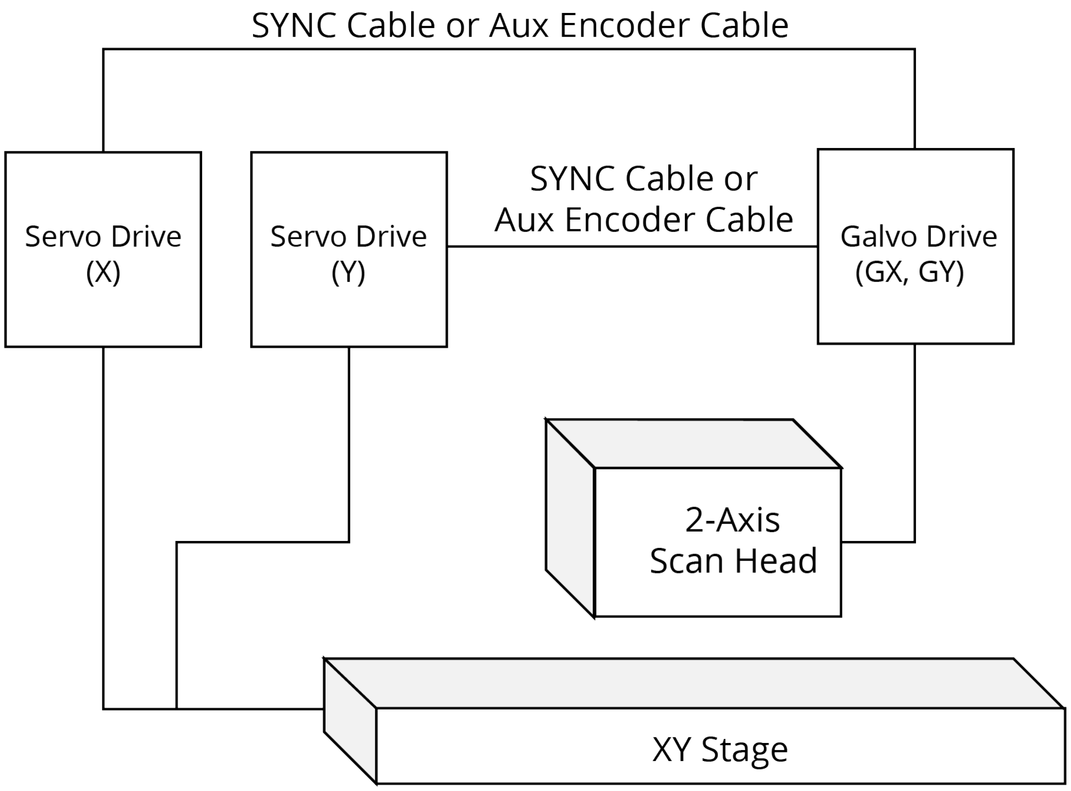


Figure 3-1: IFOV System with Cable Connections Between Drives

3.5.2. SYNC Cable Connection

HARDWARE: XC2 and XC2e drives do not have SYNC ports. Thus, you must use Auxiliary Encoder cables with flying leads for this step. Make sure that your selected drive hardware supports one of the configurations that follow and has the correct cable assemblies.

The SYNC cable is a proprietary serial communication link from Aerotech that uses a USB-3.0 Type A style connector. Many Aerotech Automation1 servo drives and galvo scanner drives have SYNC ports as standard connectivity features. These drives have two SYNC cable inputs labeled **SYNC A** and **B**, which are arbitrary and can be assigned in software configuration to a specific axis input or output. This is shown in Figure 3-2. The cables create a single connection between collinear axis pairs.

For Example

Collinear axis pair **SX** and **GX** would have a SYNC cable connected between the Automation1 GL4 **SYNC A** and the Automation1 XC4e **SYNC A** ports for the **SX** axis.

The figure that follows shows the SYNC cable inputs on the tops of an Automation1 XC4e and an Automation1 GL4.

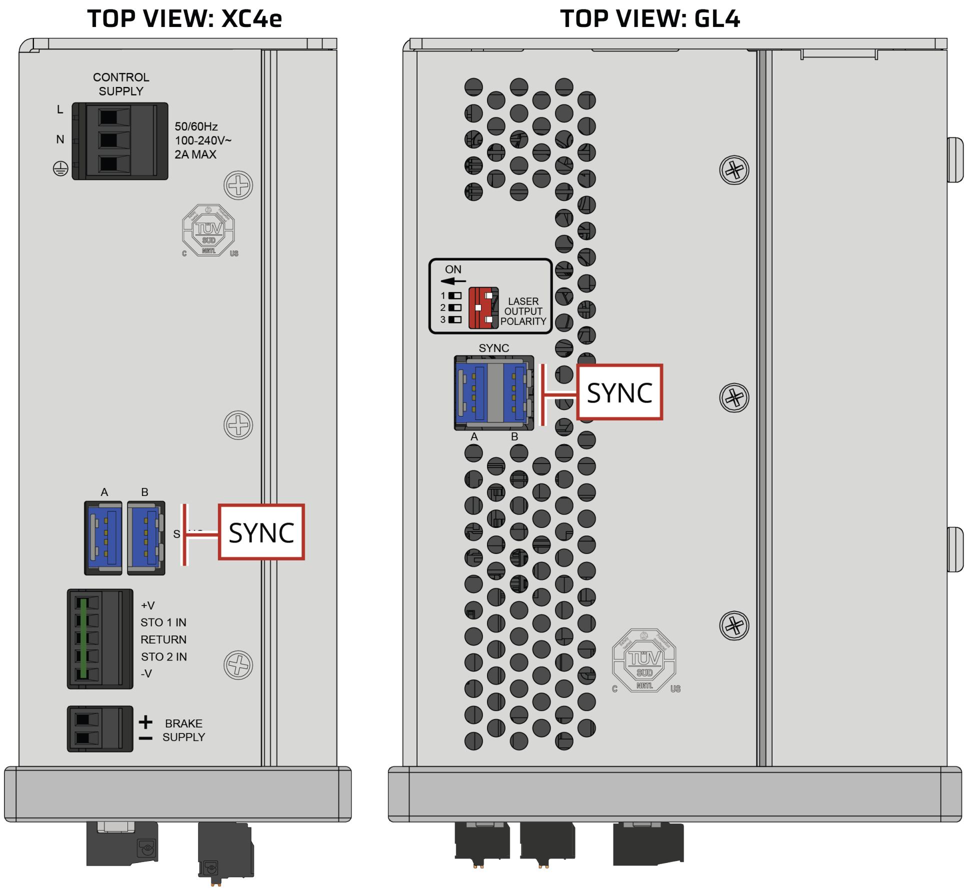


Figure 3-2: SYNC Inputs for the XC4e and GL4

Auxiliary Encoder Cable

The Auxiliary encoder output cable is typically a standard D-sub serial communication cable, either standard or small size. This cable has flying leads on the opposite end that you must wire into the applicable **Sin+**, **Sin-**, **Cos+**, **Cos-** terminals on the Automation1 GL4 input terminals.

HARDWARE: For specific pin assignments on these cables and the pin locations on the Automation1 GL4 input terminals (as shown in Figure 3-3), refer to the hardware manual for your Automation1 servo drive. To download your manual from www.aerotech.com, go to the **Automation1** section of Manuals & Help Files.

The figure that follows shows the **Encoder Input** terminals on the front of an Automation1 GL4.

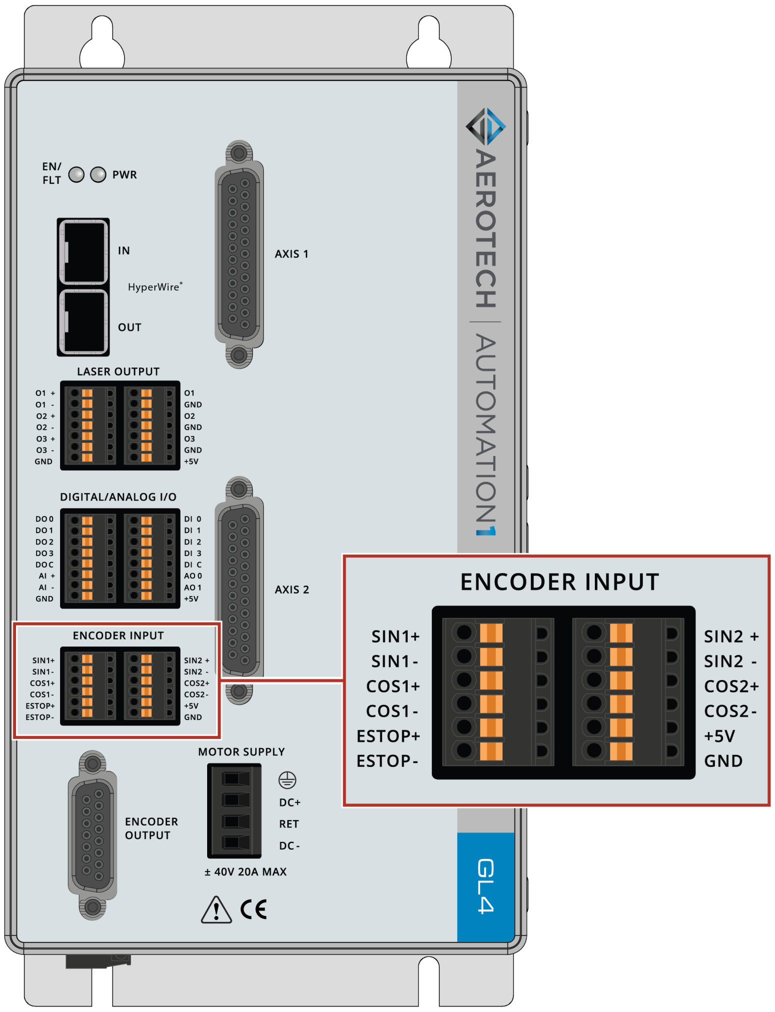


Figure 3-3: Automation1 GL4 Encoder Input Terminals

HARDWARE: For more information about the pin assignments for inputs and outputs on the Automation1 drives, refer to the drive hardware manuals. To download your manual from www.aerotech.com, go to the **Automation1** section of Manuals & Help Files. To access cable diagrams for specific pin assignments and wire color designations, use the Global Technical Support Portal to contact Aerotech Global Technical Support.

3.5.3. High Speed Output (HSOUT) Cable (XR3 Only)

The High Speed Output (HSOUT) on the Automation1 XR3 rack-mounted drive is a unique output mechanism that enables any of the six axes on the Automation1 XR3 to output its encoder signals to the Automation1 GL4. This cable connection is a 25-pin D-sub cable that contains individual pin assignments for sin/cos encoders. The opposite end of the cable has flying leads, which must have the correct collinear servo axis wired into the applicable **Sin+**, **Sin-**, **Cos+**, **Cos-** terminals on the Automation1 GL4 auxiliary Encoder Input terminals shown in Figure 3-3.

Chapter 4: IFOV Software Configuration

After you select, wire, and install the correct hardware, you must use Automation1 Studio and the AeroScript program file that defines and executes the IFOV motion to complete your IFOV configuration. The sections that follow help you configure all the controller parameters and connections between the servo and galvo scanner drive components. Thus, you can make sure the galvo scanner axes can track the servo axis position when IFOV is enabled.

4.1. Automation1 Studio Setup

Before you start this section, you must complete the Machine Setup to make sure all the hardware is correctly configured. This setup includes assigning the servo axes to their respective drives, configuring the Aerotech AGV galvo scanner or a third-party galvo scanner with a lens, and making sure the galvo scanner is connected to the Automation1 GL4 or Automation1 GI4 drive.

HARDWARE: When you assign a lens to the galvo hardware specified in Machine Setup, this defines the spatial relation between the angular position change of the mirrors and the linear displacement of the laser on the working surface, referred to as counts per unit. If you specify the incorrect **Effective Focal Length** (EFL) and **Field of View** (FOV) in the lens properties, this causes incorrectly scaled motion on the part surface. Before you specify the lens properties, make sure to review the lens manufacturer's specifications for EFL and FOV.

After you specify the lens properties in Machine Setup, you must make sure that the lens is spaced correctly at the scanner's output (M2 Distance) and that the scanner + lens is at the correct working distance over the part surface.

Lenses and Spacers:

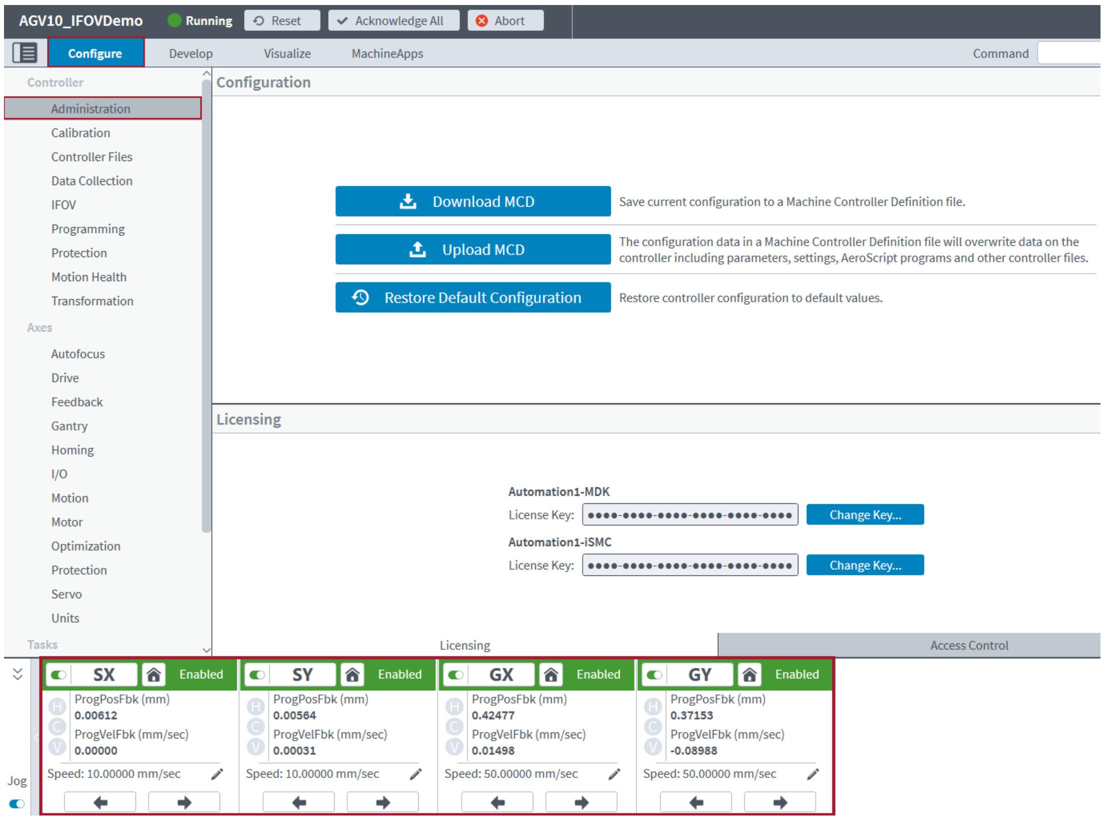
Standard lenses sold by Aerotech include pre-designed spacers. To purchase them, contact your Aerotech sales representative. See the Contact Sales section of www.aerotech.com.

For non-standard lenses, custom spacers must be designed. Aerotech can also provide support for this. Contact Sales for more information.

You can find the dimensions that are necessary for spacer design in the AGV hardware manuals. To download your manual from www.aerotech.com, go to the **Mechanical** section of Manuals & Help Files.

This section also requires you to complete the servo axis checklist items, which include homing, tuning, and other related setup items. For instructions about how to do this set of operations, see the Recommended Next Steps of Machine Setup.

Figure 4-1 shows the **SX**/**SY** linear servo axes and the **GX**/**GY** galvo scanner axes in Automation1 Studio. All the axes will home and jog correctly. If this does not occur, the axes will require more configurations that are out of the scope of this IFOV guide.



Download MCD — Save current configuration to a Machine Controller Definition file.

Upload MCD — The configuration data in a Machine Controller Definition file will overwrite data on the controller including parameters, settings, AeroScript programs and other controller files.

Restore Default Configuration — Restore controller configuration to default values.

Automation1-MDK
License Key: ••••-••••-••••-••••-••••-••••

Automation1-iSMC
License Key: ••••-••••-••••-••••-••••-••••

	SX	SY	GX	GY
ProgPosFbk (mm)	0.00612	0.00564	0.42477	0.37153
ProgVelFbk (mm/sec)	0.00000	0.00031	0.01498	-0.08988
Speed	10.00000 mm/sec	10.00000 mm/sec	50.00000 mm/sec	50.00000 mm/sec

Figure 4-1: Linear Servo (SX/SY) and Galvo Scanner (GX/GY) Axes

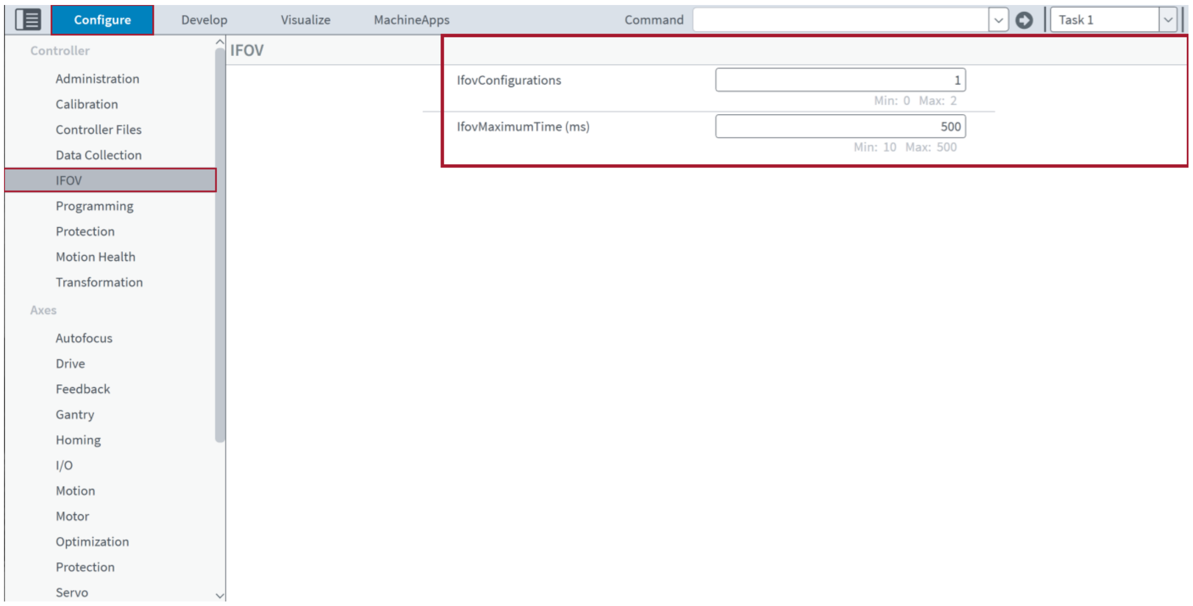
4.1.1. IFOV Configuration Setup

The first setup step is to make sure the IfovConfigurations Parameter is set with the necessary value in the **Configure** workspace of Automation1 Studio. This value is the number of IFOV configurations connected to the controller, which is either 0 (off), 1 (single galvo scanner), or 2 (multi-galvo scanner). If you set the IfovConfigurations parameter to 1, this value applies to all single galvo scanner systems and will be used in this guide.

Also set the IfovMaximumTime Parameter to the maximum value of 500 ms. This value enables maximum flexibility when you configure the `IfovSetTime()` function, which is discussed later in this guide.

Figure 4-2 shows the location of these parameters in the **Configure** workspace of Automation1 Studio and the recommended setup values for this guide. The values for these parameters are typical settings for most IFOV configurations.

IMPORTANT: You must reset the controller for these parameter changes to have an effect.



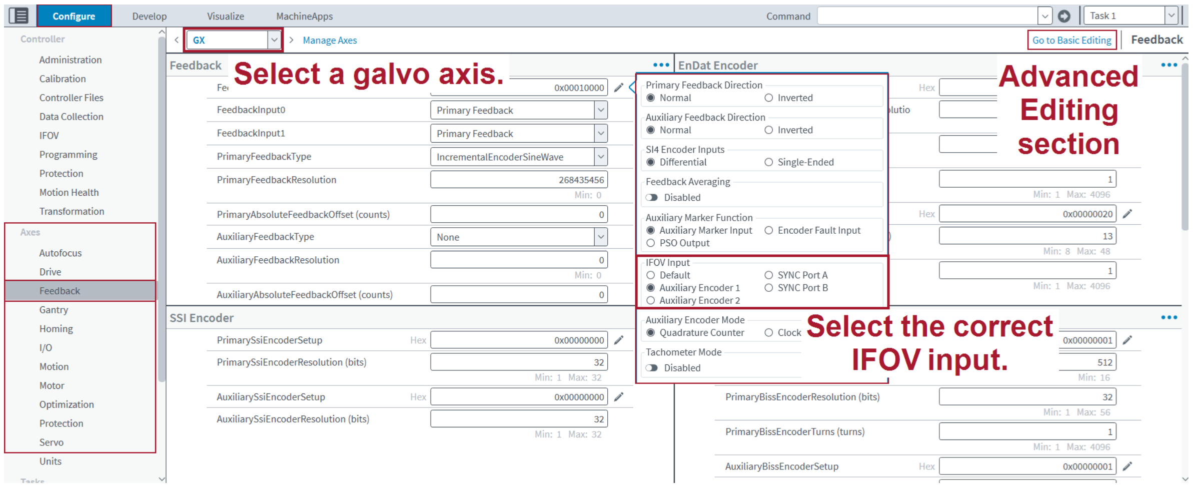
IFOV

IfovConfigurations	1	Min: 0 Max: 2
IfovMaximumTime (ms)	500	Min: 10 Max: 500

Figure 4-2: Typical Parameter Values for Most IFOV Configurations

4.1.2. Configure IFOV Feedback Inputs for Galvo Scanner Axes

Section 3.5.1. Servo Axis Encoder Wiring supplies two possible configurations to connect the servo axis feedback to the galvo scanner axis drive. You can use SYNC cables or manually wire the encoder signals through the Auxiliary Encoder Output or High Speed Output (XR3 only) cables. Then you must change the galvo scanner axis feedback setup mask to show where the servo encoder signals will enter the GL4 or GI4 drive. For each galvo scanner axis (**GX**/**GY**), the feedback input mask must be defined to match the physical hardware wiring. Figure 4-3 shows you where to find the **Advanced Editing** section of the **Feedback** topic in Automation1 Studio. It also shows you how to configure the FeedbackSetup mask to make the **IFOV Input** match the physical wiring of the system. Refer to the procedure that follows for more information.



Feedback

Fe	0x00010000	
FeedbackInput0	Primary Feedback	
FeedbackInput1	Primary Feedback	
PrimaryFeedbackType	IncrementalEncoderSineWave	
PrimaryFeedbackResolution	268435456	Min: 0
PrimaryAbsoluteFeedbackOffset (counts)	0	
AuxiliaryFeedbackType	None	
AuxiliaryFeedbackResolution	0	Min: 0
AuxiliaryAbsoluteFeedbackOffset (counts)	0	

SSI Encoder

PrimarySsiEncoderSetup (Hex)	0x00000000	
PrimarySsiEncoderResolution (bits)	32	Min: 1 Max: 32
AuxiliarySsiEncoderSetup (Hex)	0x00000000	
AuxiliarySsiEncoderResolution (bits)	32	Min: 1 Max: 32

EnDat Encoder

Primary Feedback Direction: [✓ Normal] Inverted
Auxiliary Feedback Direction: [✓ Normal] Inverted
SI4 Encoder Inputs: [✓ Differential] Single-Ended
Feedback Averaging: Disabled
Auxiliary Marker Function: [✓ Auxiliary Marker Input] Encoder Fault Input, PSO Output
IFOV Input: Default, SYNC Port A, [✓ Auxiliary Encoder 1], SYNC Port B, Auxiliary Encoder 2
Auxiliary Encoder Mode: [✓ Quadrature Counter] Clock
Tachometer Mode: Disabled

Hex		
lutio		
	1	Min: 1 Max: 4096
Hex	0x00000020	
	13	Min: 8 Max: 48
	1	Min: 1 Max: 4096
	0x00000001	
	512	Min: 16
PrimaryBissEncoderResolution (bits)	32	Min: 1 Max: 56
PrimaryBissEncoderTurns (turns)	1	Min: 1 Max: 4096
AuxiliaryBissEncoderSetup (Hex)	0x00000001	

Figure 4-3: IFOV Configuration for FeedbackSetup Mask

How to set the IFOV Input axes to read the correct signals

1. Open Automation1 Studio.
2. At the top, select the **Configure** tab to go to the **Configure** workspace.
3. On the left panel, find the **Axes** category. Then select the **Feedback** topic.
4. In the upper right corner, click **Go to Advanced Editing**.

5. In the upper left corner, find the Axis selector drop-down. Use it to select one of the galvo scanner axes.
6. In the **Feedback** module, find the FeedbackSetup Parameter. Click the pencil icon next to this parameter. A selection window comes into view.
7. For the **IFOV Input** setting, specify the correct input for the galvo scanner axis that you selected.

> **Tip**: In step **7**, the correct SYNC port or Auxiliary Encoder must be selected for the axis specified in step **5**. In this example, **GX** is the primary axis for Automation1 GL4 in Machine Setup, thus **SYNC Port A** or **Auxiliary Encoder 1** must be selected. This is the default configuration from the Aerotech factory. If you are not sure, re-enter Machine Setup and verify which axis name is associated with Automation1 GL4 Axis #1.

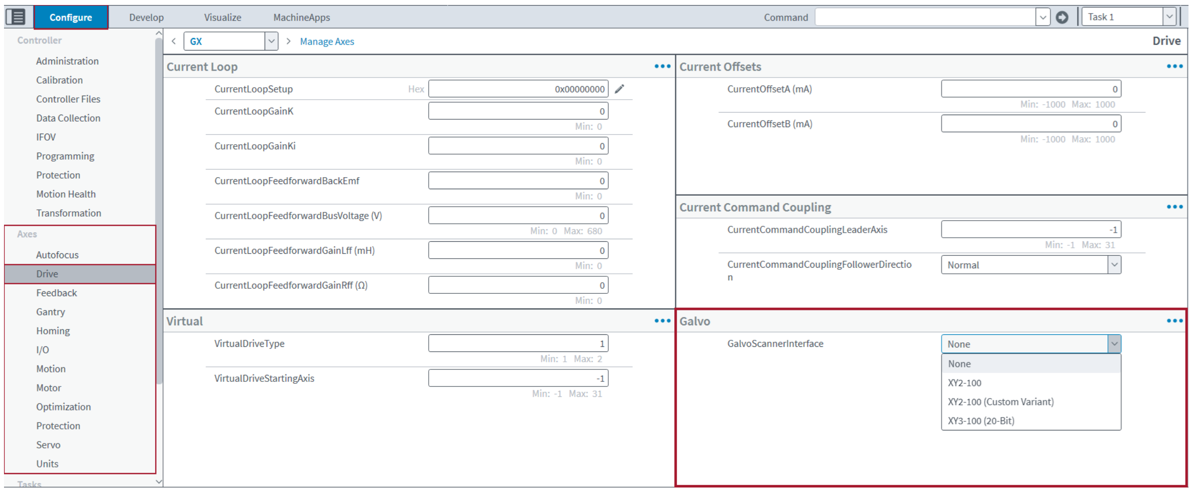
8. Repeat steps **5 - 7** for the other galvo scanner axis. Then continue to the next step.
9. Click the **Save All** button to save your parameter changes.
10. Reset the controller for these changes to have an effect.

4.1.3. Automation1 GI4 Setup Parameters

The Automation1 GI4 galvo scanner interface converts the trajectory command (which is usually sent to the Automation1 GL4 and AGV 2-axis galvo scanner) into the XY2-100 or XY3-100 protocol. For the system to operate correctly, you must specify the galvo scanner trajectory conversion. Figure 4-4 shows the location of this setting in Automation1 Studio. Refer to the procedure that follows for more information.

How to select a third-party galvo scanner protocol for the Automation1 GI4

1. Open Automation1 Studio.
2. At the top, select the **Configure** tab to go to the **Configure** workspace.
3. On the left panel, find the **Axes** category. Then select the **Drive** topic.
4. In the **Galvo** module, find the GalvoScannerInterface Parameter. Click the drop-down arrow and select one of the galvo scanner protocols to output.
5. Click the **Save All** button to save your parameter changes.
6. Reset the controller for these changes to have an effect.



Current Loop

CurrentLoopSetup	Hex 0x00000000	
CurrentLoopGainK	0	Min: 0
CurrentLoopGainKi	0	Min: 0
CurrentLoopFeedforwardBackEmf	0	Min: 0
CurrentLoopFeedforwardBusVoltage (V)	0	Min: 0 Max: 680
CurrentLoopFeedforwardGainLff (mH)	0	Min: 0
CurrentLoopFeedforwardGainRff (Ω)	0	Min: 0

Virtual

VirtualDriveType	1	Min: 1 Max: 2
VirtualDriveStartingAxis	-1	Min: -1 Max: 31

Current Offsets

CurrentOffsetA (mA)	0	Min: -1000 Max: 1000
CurrentOffsetB (mA)	0	Min: -1000 Max: 1000

Current Command Coupling

CurrentCommandCouplingLeaderAxis	-1	Min: -1 Max: 31
CurrentCommandCouplingFollowerDirection	Normal	

Galvo

GalvoScannerInterface	None

None
XY2-100
XY2-100 (Custom Variant)
XY3-100 (20-Bit)

Figure 4-4: Select a Third-Party Galvo Scanner Protocol to Output from Automation1 GI4

4.1.4. Task-Specific Parameters

For IFOV to operate correctly on a specified task, you must define values for some task-specific motion parameters. As a best practice, change these values for each task (1 - 32) where IFOV motion will occur. To change the task-specific motion parameters, refer to the procedure that follows.

How to change the task-specific parameters

1. Open Automation1 Studio.
2. At the top, select the **Configure** tab to go to the **Configure** workspace.
3. On the left panel, find the **Tasks** category. Then select the **Motion** topic. For more information about this topic, see Motion Topic (Tasks Category).
4. In the **Lookahead Motion** module, set the MaxLookaheadMoves Parameter to 200 or greater.
5. In the **Trajectory Generation** module, set the MotionUpdateRate Parameter to 100 kHz.

   IMPORTANT: Make sure you change the values for these parameters on each task (1 - 32) where IFOV motion will occur.

6. Click the **Save All** button to save your parameter changes.
7. Reset the controller for these changes to have an effect.

4.1.5. Testing and Correcting Collinear Axis Motion Direction

Tip: Aerotech recommends that you project a targeting laser through the input aperture of the galvo scan head so you can verify the direction of positive motion for each axis pair.

HARDWARE: This section requires the horizontal and vertical servo and galvo axis pairs to be collinear with each other and the matching coordinate system. Before you continue with this section, you must make sure this is configured correctly. See Section 3.5. Cables and Wiring for more information.

HARDWARE: Servo and Galvo axes sold by Aerotech have default positive motion directions defined in their user manuals. This factory-defined positive motion direction is directly related to the axis encoder's home marker position and which direction of travel results in a positive count change versus a negative count change. You can find these motion directions in the drive hardware manuals. To download these manuals from www.aerotech.com, go to the **Mechanical** section of Manuals & Help Files.

Make sure that you know these default orientations relative to the global coordinate system of the machine. If you do not know this information and do not correctly set the ReverseMotionDirection Parameter for each axis, this will have an unsatisfactory effect on trajectory programming and how you interact with jog behavior.

As the user, you determine the positive motion direction within the global coordinate system of the machine. Figure 4-5 shows the demonstration system of this IFOV guide, where the servo axes (**SX** and **SY**) and the galvo scanner axes (**GX** and **GY**) match the positive motion direction of the global coordinate system.

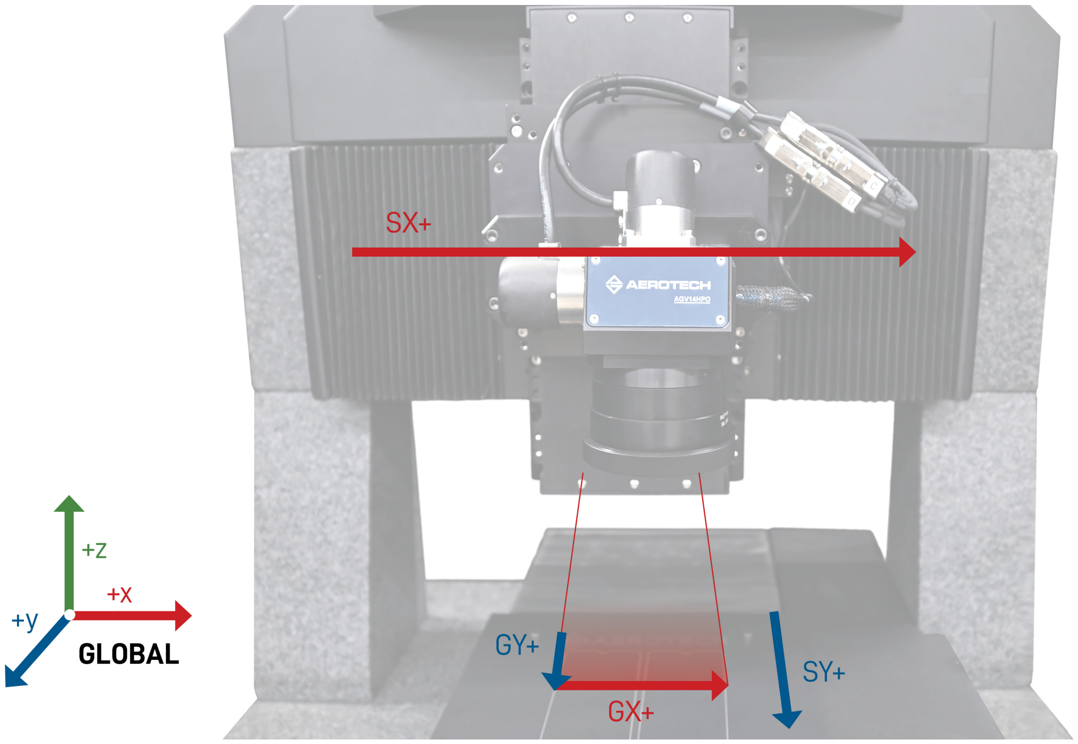


Figure 4-5: IFOV Demonstration with Global Coordinate System

The default setup is one where the ReverseMotionDirection Parameter is disabled for all axes. After you run the motion tests below to determine each axis's positive motion direction, you must verify that each collinear axis pair (e.g., **SX** and **GX**) is configured so that a positive (+) jog command moves the laser spot on the part's surface in the same physical direction. If the test shows that a collinear axis pair does not move the laser spot in the same direction, you must enable the ReverseMotionDirection parameter on one of the two axes to make sure the servo axis tracking is correct for IFOV.

Do a test to see if the positive direction of motion for each collinear axis pair is set up correctly. To do this, use the jog buttons for each axis to verify some conditions. Then examine the direction of motion to which the laser pointer moves on the part surface.

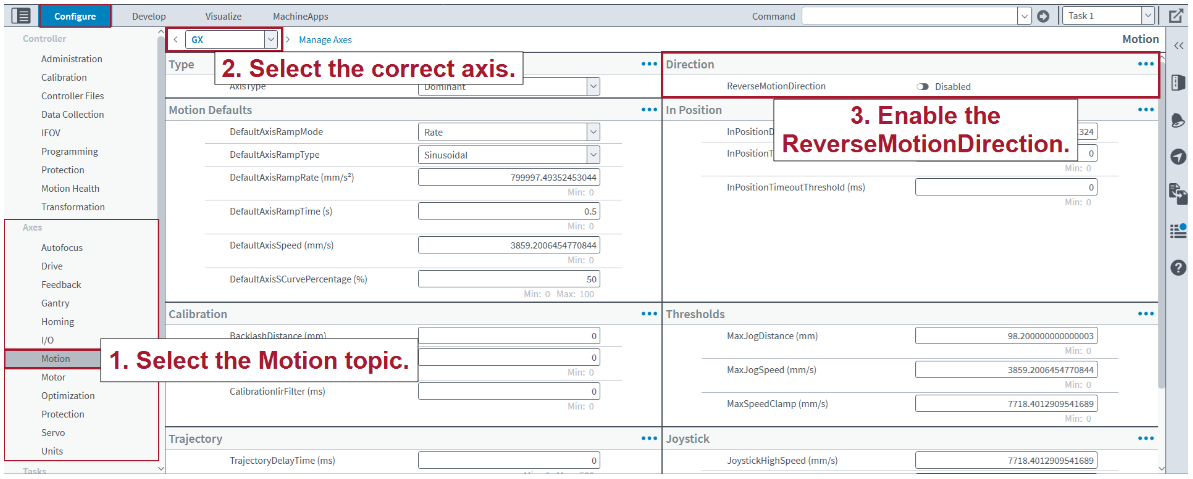
How to test the positive motion direction of each collinear axis pair

1. Jog the servo **X** axis in the positive direction. Then do the steps that follow:
   A. Record which direction the laser pointer moves on the part surface.
   B. Jog the galvo scanner **X** axis in the positive direction.
   C. Record which direction the laser pointer moves on the part surface.
   D. For the two axes that you jog, make sure the laser pointer moves in the same direction on the workpiece and aligns with the positive **X** direction of the global machine.
2. Jog the servo **Y** axis in the positive direction. Then do the steps that follow:
   A. Record which direction the laser pointer moves on the part surface.
   B. Jog the galvo scanner **Y** axis in the positive direction.
   C. Record which direction the laser pointer moves on the part surface.
   D. For the two axes that you jog, make sure the laser pointer moves in the same direction on the workpiece and aligns with the positive **Y** direction of the global machine.

If you do the test and cannot verify one or both of these conditions, refer to Figure 4-6 and the How to invert any single axis direction procedure to reverse the direction of any single axis or a set of axes to make sure the two axes move the laser pointer in the same direction. Also, make sure these positive motion directions align with the positive direction of the global machine.

For Example

If the servo coordinates are aligned with the machine coordinate system but only the galvo scanner **X** axis is reversed, refer to the How to invert any single axis direction procedure to invert the necessary axes. Make sure to align all the axes with respect to the global machine coordinates. This might occur if you mount the galvo scanner in a non-standard orientation with respect to the servo stages (e.g., rotated 90 degrees about the **Z** axis).



Type
AxisType: Dominant

Motion Defaults
DefaultAxisRampMode: Rate
DefaultAxisRampType: Sinusoidal
DefaultAxisRampRate (mm/s²): 799997.49352453044 (Min: 0)
DefaultAxisRampTime (s): 0.5 (Min: 0)
DefaultAxisSpeed (mm/s): 3859.2006454770844 (Min: 0)
DefaultAxisSCurvePercentage (%): 50 (Min: 0 Max: 100)

Calibration
BacklashDistance (mm): 0
0 (Min: 0)
CalibrationIirFilter (ms): 0 (Min: 0)

Trajectory
TrajectoryDelayTime (ms): 0

Direction
ReverseMotionDirection: Disabled

In Position
InPositionD...: ...324
InPositionT...: 0 (Min: 0)
InPositionTimeoutThreshold (ms): 0 (Min: 0)

Thresholds
MaxJogDistance (mm): 98.200000000000003 (Min: 0)
MaxJogSpeed (mm/s): 3859.2006454770844 (Min: 0)
MaxSpeedClamp (mm/s): 7718.4012909541689 (Min: 0)

Joystick
JoystickHighSpeed (mm/s): 7718.4012909541689

Figure 4-6: Invert Axis Direction in Automation1 Studio

How to invert any single axis direction

1. Open Automation1 Studio.
2. At the top, select the **Configure** tab to go to the **Configure** workspace.
3. On the left panel, find the **Axes** category. Then select the **Motion** topic.
4. In the upper left corner of the application, record the axis that is selected. Then select the axis for which the direction must be inverted.
5. Enable the ReverseMotionDirection Parameter setting to invert the single axis.
6. Reset the controller for these changes to have an effect.

4.2. AeroScript Commands Required for IFOV Setup

For IFOV to operate correctly with any AeroScript program, you must issue a group of AeroScript setup commands. Use one of the methods that follow:

A. Issue the commands inside the primary AeroScript program (`.ascript`) where motion is executed. This method requires any program that uses IFOV to include all the commands. You can put these commands in one of the locations that follow. You can either put them at the top of the program before you issue the **`IfovOn()`** function or in a user-defined task function at the bottom of the program to keep it more organized. For more information about how to do this, see User-Defined Functions in the **AeroScript Programming** section of Automation1 Help. Also see IFOV Functions for information about the IFOV functions.

B. Issue the commands in an AeroScript library function within an AeroScript library (`.ascriptlib`) file. After you set up this file, you can configure it within controller automation and invoke it as a function within any task or program on the controller. For information about how to do this, see Libraries in the **AeroScript Programming** section of Automation1 Help.

This guide will focus on method **B**. You can use a different setup method for IFOV if necessary. The full library file and its method of operation are included in Section 8.1. IFOV-Setup-Library Program. You can copy, configure, and save the file to the controller to enable easy single-line setup of IFOV.

IMPORTANT: You must issue the functions shown in Section 4.2.2. Defining Axis Variables for Repeatable Setup before you issue the **`IfovOn()`** function to turn on IFOV. Also, you must issue the **`IfovOff()`** function before you issue a different set of functions to change the IFOV parameters.

4.2.1. Turning IFOV On and Off

You can turn IFOV on and off at specific locations within a program, which lets you define motion of the servo and galvo scanner axes with or without IFOV turned on. You might also want to turn off IFOV when you do homing, offsetting, or other application-specific motion with only the galvo scanner or servo axes at a time. To turn IFOV on and off, use the functions that follow:

- **`IfovOn()`**
- **`IfovOff()`**

Issue the IFOV on and off functions only after you issue all the setup code, which is discussed in the sections that follow. If you do not complete the setup steps or you issue them after the **`IfovOn()`** function, this can cause errors or unwanted behavior. For more information about these functions, see IFOV Functions.

4.2.2. Defining Axis Variables for Repeatable Setup

The sections that follow show the AeroScript functions in the order they must be executed in a program. There are also descriptions of what each function does and why it is necessary. To keep the code snippets small and readable, the code snippet that follows is supplied. You must put this code snippet at the top of the program before all the other functions to make sure they operate correctly.

Tip: If you copy this code snippet from the PDF file, some of the formatting might be different from what you see in this guide. To see the full IFOV example program, refer to Section 8.1. IFOV-Setup-Library Program.

```
// Define the IFOV axes. By defining these
// axes at the top of your IFOV program,
// you can use them throughout the code snippets.
// Specify the horizontal galvo scanner axis.
var $HGalvo as axis = GX
// Specify the vertical galvo scanner axis.
var $VGalvo as axis = GY
// Specify the horizontal servo axis.
var $HServo as axis = SX
// Specify the vertical servo axis.
var $VServo as axis = SY

// Define the IFOV axes.
// $axisPairH is the horizontal pair (Servo & Galvo scanner).
// $axisPairV is the vertical pair (Servo & Galvo scanner).
// Define the collinear horizontal pair for IFOV setup.
var $axisPairH[] as axis = [$HGalvo, $HServo]
// Define the collinear vertical pair for IFOV setup.
var $axisPairV[] as axis = [$VGalvo, $VServo]
// Defines all the IFOV axes (does not include the Z axis).
var $allAxes[] as axis = [$HGalvo, $HServo, $VGalvo, $VServo]
```

4.2.3. Defining IFOV Parameters and Associated Variables

After you define the axis variables for setup, you must define and configure the axis variables that are used during motion. This section will make sure that axis and coordinated motion are configured to use the correct process speeds. You must define the settings that follow as part of this configuration.

1. **Coordinated Speed and Acceleration** - The speed at which the laser spot will move during coordinated motion commands (e.g., `G1` or `MoveLinear()`) on the surface of the workpiece and the maximum permitted acceleration to get to that speed.
2. **Galvo Scanner Axis Speed and Acceleration** - The maximum permitted speed and acceleration of the galvo scanner axes during IFOV motion. This applies to all non-coordinated motion, such as `G0` or `MoveRapid()`. Also see Motion Setup Functions for `SetupAxisSpeed()`, `SetupAxisRampType()`, and `SetupAxisRampValue()`.
3. **Servo Axis Tracking Speed and Acceleration** - The maximum permitted speed (`IfovSetTrackingSpeed()`) and acceleration (`IfovSetTrackingAcceleration()`) of the servo axis during IFOV motion.
4. **IFOV FOV Size** - The `IfovSetSize()` function defines the maximum field of view (FOV) for the galvo scanner to operate within during IFOV motion. The FOV that you specify can be any size smaller than the actual galvo scanner FOV up to the full FOV size, but it cannot be larger.
5. **IFOV Search Time** - The `IfovSetTime()` function defines the lookahead time on which IFOV operates. It must use a value between 10 and 500 ms. A lower value will decrease galvo scanner motion and increase servo motion. A higher value will increase galvo scanner motion and decrease servo motion.
6. **More Synchronized Axes with IFOV** - The `IfovSetSyncAxes()` function enables a maximum of two more axes that will be synchronized with IFOV motion. Specified axes will have specific AeroScript functions, such as motion and I/O synchronized with IFOV motion.
7. **IFOV Time Granularity** - Use the `IfovSetup()` function to set the $IfovTimeGranularity value. This value defines the update rate at which IFOV operates. The default value is 5 ms. You can also set it to 1 ms. If you decrease this value, it will increase the CPU load on the controller. But it will also enable a

higher maximum permitted galvo scanner speed and a higher permitted coordinated speed. If you do not set this value correctly, a task error might occur.

- If you set this value to 5, the maximum permitted speed = 100 x FOV Size.

For Example

For a 10 mm FOV, the maximum permitted speed = 1,000 mm/s.

- If you set this value to 1, the maximum permitted speed = 1000 x (½ x FOV Size).

For Example

For a 10 mm FOV, the maximum permitted speed = 5,000 mm/s.

To set the task-based acceleration, see the setup information about ramp rates in the Ramp Mode section of Motion Setup Functions. For IFOV, rate-based ramping is the default configuration and Aerotech recommends that you use a large ramp rate for the best IFOV performance. You can set the ramp rate to 0 for an infinite ramp rate.

For axis-based acceleration, use a large ramp rate. Also refer to the Ramp Mode section of Motion Setup Functions.

When you use IFOV mode, you must set the speed and acceleration rates for coordinated motion. The controller will automatically limit the speed and acceleration rates for servo and galvo scanner axes to get the necessary coordinated speed or laser-on-workpiece speed. This makes the setup for IFOV easier because you only need to think about how fast the laser spot must move and the maximum permitted acceleration necessary to enable that. You do not need to consider the capability of each axis.

After you specify the speeds and accelerations, use the AeroScript code that follows to set these values. This AeroScript code snippet will define variables that are easier to edit at the top of a program. Then it will define the applicable setup commands for the IFOV configuration commands. You must include these defined variables and setup commands in any program where IFOV is enabled for it to operate correctly.

Tip: If you copy this code snippet from the PDF file, some of the formatting might be different from what you see in this guide. To see the full IFOV example program, refer to Section 8.1. IFOV-Setup-Library Program.

IMPORTANT: The AeroScript code sample that follows spans two pages.

```
// Define and configure the maximum permitted galvo speed.
// Time Granularity (must be 1 or 5) is a value of time in ms that
// defines the minimum update rate for the IFOV algorithm to do
// its operations. The default value is 5 ms and
// The minimum value is 1 ms.
var $IfovTimeGranularity as integer = 1
// Set $IfovTimeGranularity to the specified value.
IfovSetup(IfovSetupItem.TimeGranularity, $IfovTimeGranularity)

// Variable that sets the maximum galvo speed
// that is permitted during IFOV operations.
var $GalvoSpeedSet as real

    if ($GalvoSpeed == 0)
        $GalvoSpeedSet = (500*$FOVSize)/$IfovTimeGranularity
    else
        $GalvoSpeedSet = $GalvoSpeed
    end

// Set the IFOV commands for maximum speeds, accelerations,
// FOV size, search time, and tracking.
// Configures the Coordinated Ramp Type for how the
// laser will move on the part surface.
SetupCoordinatedRampType(RampType.Sine)
// Configures the Coordinated Ramp Rate for the
// laser spot on the part surface.
SetupCoordinatedRampValue(RampMode.Rate, $IfovAccel)
// Configures the Coordinated Speed for the
// laser spot on the part surface.
SetupCoordinatedSpeed($IfovSpeed)
//***When IFOV is turned on, you must specify the
// galvo ramp and speeds. IFOV will igore the
// servo axis ramp and speeds.
// Specify the ramp type for the galvo axes.
SetupAxisRampType([$HGalvo, $VGalvo], RampType.Sine)
// Specify the target ramp rate for the galvo axes.
SetupAxisRampValue([$HGalvo, $VGalvo], RampMode.Rate, $GalvoAccel)
// Specify the target speed for the galvo axes.
SetupAxisSpeed([$HGalvo, $VGalvo], [$GalvoSpeedSet, $GalvoSpeedSet])
// Configures the maximum search time in ms. Start with 200.
IfovSetTime($SearchTime)
// Configures the maximum speed of servo axes to
// track during IFOV motion.
IfovSetTrackingSpeed($ServoTrackSpeed)
// Configure the maximum ramp rate of the
// servo axes to track during IFOV.
IfovSetTrackingAcceleration($ServoTrackAccel)
// An empty list is passed, which means
// that no additional axes are synchronized with IFOV motion.
// If you want to synchronize more axes
// with IFOV motion, pass them to the function.
IfovSetSyncAxes([])
// Configures the IFOV field of view (FOV) size.
IfovSetSize($FOVSize)
```

4.2.4. Make Sure Encoder Data Rates Are Not Exceeded

The data rate of each servo encoder-channel signal that is output to the galvo-scanner encoder input must not exceed the maximum permitted data rate of 25 MHz. This is an electrical signal limitation between the servo and galvo scanner drives. You must also include any jitter in the signal. Thus, Aerotech recommends that you use a maximum data rate of 22 MHz or less.

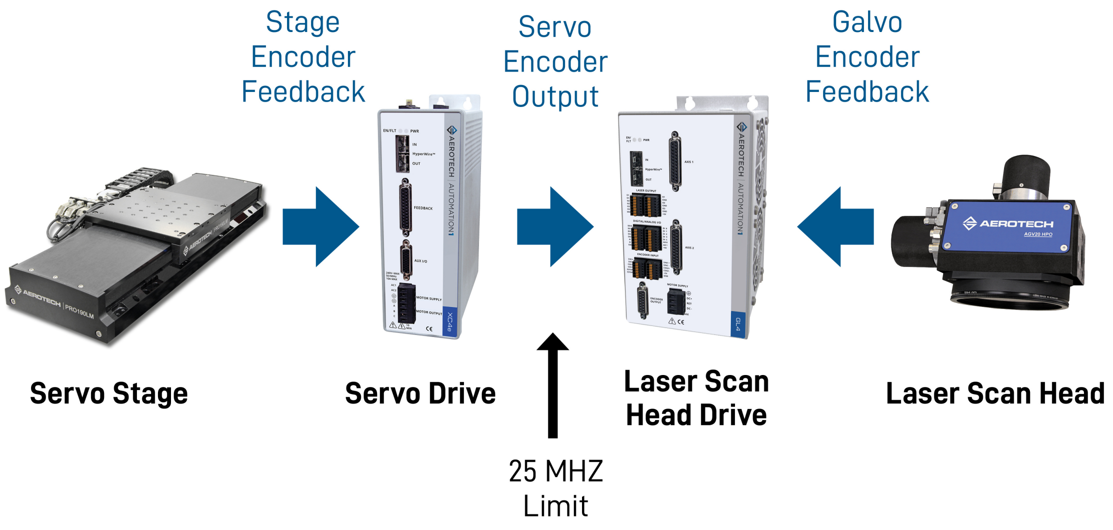


Figure 4-7: Encoder Channel Signal Schematic

To calculate the maximum encoder-channel frequency output for each servo axis to the corresponding galvo scanner axis, use the equation that follows:

$$\begin{array}{c}\text{Servo Stage Maximum}\\\text{Encoder Channel Frequency}\\\text{(Counts/S)}\end{array} = \begin{array}{c}\text{Maximum Permitted}\\\text{Servo Stage Velocity}\\\text{(Units/S)}\end{array} \times \begin{array}{c}\text{Servo Stage}\\\text{CountsPerUnit}\\\text{(Counts/Unit)}\end{array}$$

Equation: 4-1: Encoder Channel Frequency

The example that follows shows the AeroScript code snippet for this calculation:

Tip: If you copy this code snippet from the PDF file, some of the formatting might be different from what you see in this guide. To see the full IFOV example program, refer to Section 8.1. IFOV-Setup-Library Program.

```
// User input value for maximum servo speed (mm/s).
// If you use the library or function within a program,
// this variable definition will be part of the function call.
// This code is supplied as an example.
// It will be implemented differently in the
// final example at the end of this guide.
var $ServoTrackSpeed as real = 100

// Calculate the maximum encoder channel frequency of each servo stage.
// This is the highest data rate that occurs when
// each servo stage gets to the set $ServoTrackSpeed variable.
var $HServoMaxDat as real = ($ServoTrackSpeed) * (ParameterGetAxisValue
($HServo, AxisParameter.CountsPerUnit)
var $VServoMaxDat as real = ($ServoTrackSpeed) * (ParameterGetAxisValue
($VServo, AxisParameter.CountsPerUnit)
```

Now that you calculated the maximum data output rates for the servo axes at the specified maximum tracking speed, you must make sure they are correct. An invalid data rate is possible because at the specified speed the encoder channel frequency might exceed the maximum permitted value of 25 MHz. If this occurs, you must divide the encoder signal down to prevent errors. To calculate which value to use as the divider value, use the equation that follows:

$$\text{Divider Value} = \frac{\text{Axis Maximum Data Rate (Counts/S)}}{\text{Maximum Permitted Data Rate (Counts/S)}}$$

Equation: 4-2: Divider Value

The code snippet that follows does these calculations. You can use it in an AeroScript program or library.

Tip: If you copy this code snippet from the PDF file, some of the formatting might be different from what you see in this guide. To see the full IFOV example program, refer to Section 8.1. IFOV-Setup-Library Program.

```
// The maximum data rate output for all Automation1 servo drives is 25 MHz.
// Set the data rate to 22 MHz to include Jitter.
var $MaxDataRate as integer = 22000000
// Calculates the required EmulatedQuadratureDivider for $HServo or the X axis
// based on the maximum data rate at the servo tracking speed
// that you specified.
var $HDivider as real = Abs($HServoMaxDat / $MaxDataRate)
// Calculates the required EmulatedQuadratureDivider for $VServo or the Y axis
// based on the maximum data rate at the servo tracking speed
// that you specified.
var $VDivider as real = Abs($VServoMaxDat / $MaxDataRate)
```

HARDWARE: If your servo drives support the hardware multiplication of encoder signals (-CT2 or -CT4 for XR3 drives), use the PrimaryEmulatedQuadratureDivider Parameter to specify the divider value for the encoder output signal.

For servo drives that use -MX2 or -MX3 feedback (which output only square-wave digital signals), you can use the **`DriveEncoderOutputConfigureDivider()`** function to apply an integer divider to the encoder feedback signals. See Configuring the Drive Encoder Output for more information.

After you calculate the divider value, make sure this value is correct before it is written to the controller parameter. The divider value must obey the requirements that follow:

- **Divider Value must be 1 or greater -** The calculated maximum data rate for each axis divided by the maximum data rate of 22 MHz is greater than 1. If not, you must set it to 1. This value becomes the PrimaryEmulatedQuadratureDivider Parameter, which divides the encoder signal down to make sure it is below the maximum data rate.

  For Example

  A maximum tracking speed is set to 100 mm/s on a stage that has a CountsPerUnit value of 250,000 counts/mm. The maximum frequency will be 25 MHz, which is greater than 22 MHz. Thus, you must use a divider value of 2 or more.

  For Example

  A maximum tracking speed is set to 50 mm/s on a stage that has a CountsPerUnit value of 250,000 counts/mm. The maximum frequency will be 12.5 MHz. Thus, a divider value of 1 is satisfactory for you to use.

- **Quadrature Multiplication Factor divided by the PrimaryEmulatedQuadratureDivider parameter must be a whole integer -** When you divide the value of the PrimaryEncoderMultiplicationFactor Parameter by the divider value that you calculated with Equation: 4-2, the result must be a whole integer number. It is possible to specify a processing speed that does not divide evenly. But this speed will cause unexpected behavior because the drive can only generate a whole number of output counts per encoder cycle. Also, the drive can only generate a minimum of four output counts per encoder cycle. To make sure the divider value obeys the requirements, it might be necessary for you to increase the value of the PrimaryEmulatedQuadratureDivider Parameter.

For Example

A maximum tracking speed is set to 120.5 mm/s on a stage that has a CountsPerUnit of 250,000 counts/mm. The maximum encoder frequency will be 30.125 MHz. Use Equation: 4-2 to set the PrimaryEmulatedQuadratureDivider parameter to 1.3693. The multiplication factor of the stage is 5000, which means the quadrature multiplication factor will equal 3651.5007. This is not a whole integer. Thus, you must increase the PrimaryEmulatedQuadratureDivider parameter so that it completes the requirement. Then use the next smallest whole integer number (i.e., 3651) for the quadrature multiplication factor to calculate the required divider of 1.3695.

The AeroScript code snippet that follows does the checks for these two requirements and makes sure that the PrimaryEmulatedQuadratureDivider parameter is set correctly based on the specified maximum tracking speed and the configurations used for the stages.

Tip: If you copy this code snippet from the PDF file, some of the formatting might be different from what you see in this guide. To see the full IFOV example program, refer to Section 8.1. IFOV-Setup-Library Program.

IMPORTANT: The AeroScript code sample that follows spans two pages.

```
// The if statements that follow do a check to make sure that
// the PrimaryEmulatedQuadratureDivider parameter value obeys specific
// requirements before you set it.
    // The parameter value cannot be less than 1.
    // If it is less than 1, you must clamp it to 1.
    // This is a requirement.
    if $HDivider < 1
        // If the parameter value is less than 1, set the
        // PrimaryEmulatedQuadratureDivider parameter equal to 1.
        ParameterSetAxisValue
        ($HServo, AxisParameter.PrimaryEmulatedQuadratureDivider, 1)

    else
        // If > 1, do a check to see if
        // the (PrimaryEncoderMultiplicationFactor /
        // PrimaryEmulatedQuadratureDivider) is an integer.
        var $HSearch as integer = Trunc(Ceil($HDivider))
        var $HMultiFactor as integer = Trunc(ParameterGetAxisValue($HServo,
        AxisParameter.PrimaryEncoderMultiplicationFactor))
        while (($HMultiFactor / $HSearch * $HSearch != $HMultiFactor)
                $HSearch++
                if ($HSearch >= $HMultiFactor)
                    $HSearch = $HMultiFactor
                    break
                end
        end
        ParameterSetAxisValue($HServo, AxisParameter.
        PrimaryEmulatedQuadratureDivider, $HSearch)
    end

    // Cannot be less than 1. If so, clamp it to 1 (requirement).
    if $VDivider < 1
        // If < 1, set PrimaryEmulatedQuadratureDivider equal to 1.
        ParameterSetAxisValue($VServo, AxisParameter.
        PrimaryEmulatedQuadratureDivider, 1)

    else
        // If > 1, do a check to see if
        // the (PrimaryEncoderMultiplicationFactor /
        // PrimaryEmulatedQuadratureDivider)
        // is an integer.
        var $VSearch as integer = Trunc(Ceil($VDivider))
```

```
            var $VMultiFactor as integer =
            Trunc(ParameterGetAxisValue($VServo,
            AxisParameter.PrimaryEncoderMultiplicationFactor))
            while (($VMultiFactor / $VSearch) * $VSearch != $VMultiFactor)
                $VSearch++
                if($VSearch >= $VMultiFactor)
                    $VSearch = $VMultiFactor
                    break
                end
            end
            ParameterSetAxisValue($VServo,
            AxisParameter.PrimaryEmulatedQuadratureDivider, $VSearch)

        end
```

4.2.5. Configure and Define IFOV Scale Factors and Axis Pairs

After you define all the speeds and accelerations and you set the PrimaryEmulatedQuadratureDivider parameter correctly, you must define the IFOV axis pairs with the applicable scale factors. These scale factors convert the encoder counts received from the servo axes on the drive auxiliary encoder inputs of the galvo scanner. If these counts are not scaled correctly, the IFOV motion that occurs will not be accurate.

To set these IFOV axis pairs, you must issue two functions within the program. Refer to the procedure that follows.

How to set the IFOV axis pairs

1. Calculate accurate scale factors for the horizontal and vertical axis pairs. The scale factor defines the relation between the CountsPerUnit of the galvo scanner axis and servo axis. If the scale factor is not scaled correctly, the positions commanded to the galvo scanner or servo axis might be incorrect. To calculate the scale factor for each configured axis pair, use the equation that follows:

$$\text{Scale Factor} = \frac{\substack{\text{CountsPerUnit} \\ \text{of galvo scanner axis}}}{\substack{\text{CountsPerUnit} \\ \text{of servo axis}}} \times \substack{\text{PrimaryEmulatedQuadratureDivider} \\ \text{of servo axis}}$$

   Equation: 4-3: Scale Factor

2. Issue the **`IfovSetAxisPairs()`** function to define the axis pairs and their related scale factor. This function defines the axis pairs for the IFOV algorithm to use and scales the encoder signals for those axes to make sure correct tracking occurs between the collinear axis pairs. This function has four arguments. Specify them as necessary:
   - *`$axisPairH`* - Horizontal axis pair. The collinear axis pair along the horizontal axis of travel.
   - *`$axisPairV`* - Vertical axis pair. The collinear axis pair along the vertical axis of travel.
   - *`$scaleFactorH`* - Encoder scale factor for the horizontal axis pair.
   - *`$scaleFactorV`* - Encoder scale factor for the vertical axis pair.

   The **`IfovSetAxisPairs()`** function maps the two galvo scanner axes to the two servo axes. If you do not specify encoder scale factors as the third and fourth arguments, the controller automatically calculates the scale factor by dividing the CountsPerUnit of the servo axes by the CountsPerUnit of the galvo scanner axes. For more information about this function, see IFOV Functions. For more information about the CountsPerUnit parameter, see CountsPerUnit Parameter.

The AeroScript code snippet that follows defines variables for the scale factors of the horizontal and vertical axis pairs. Then it assigns the calculated scale factors to the defined axis pairs.

Tip: If you copy this code snippet from the PDF file, some of the formatting might be different from what you see in this guide. To see the full IFOV example program, refer to Section 8.1. IFOV-Setup-Library Program.

```
// Define and calculate the encoder scale factor for
// $scaleFactorH($HGalvo, $HServo) and $scaleFactorV($VGalvo, $VServo).
var $scaleFactorH as real = (ParameterGetAxisValue($HGalvo,
AxisParameter.CountsPerUnit)/ParameterGetAxisValue($HServo,
AxisParameter.CountsPerUnit))*ParameterGetAxisValue($HServo,
AxisParameter.PrimaryEmulatedQuadratureDivider)

var $scaleFactorV as real = (ParameterGetAxisValue($VGalvo,
AxisParameter.CountsPerUnit)/ParameterGetAxisValue($VServo,
AxisParameter.CountsPerUnit))*ParameterGetAxisValue($VServo,
AxisParameter.PrimaryEmulatedQuadratureDivider)

Dwell(0.01)

// Define the IFOV axis pairs with the calculated
// IFOV encoder scale factors.
IfovSetAxisPairs($axisPairH, $axisPairV, $scaleFactorH, $scaleFactorV)
```

4.2.6. Configure and Echo Servo Encoder Signals to Galvo Scanner Axes

In Section 3.5.1. Servo Axis Encoder Wiring, you made the manual connections between the collinear servo and galvo scanner axes. But you must tell the controller which signals are connected so that IFOV can use them. It is necessary to configure and turn on the encoder signals from the horizontal and vertical servo axes. To correctly configure and output the encoder signals, do the steps that follow. For the code snippets that follow, you can copy and paste them into an `.ascript` program or `.ascriptlib` library file.

How to configure and output the encoder signals

1. Configure the **`DriveEncoderOutputConfigureInput()`** function. Do the steps that follow:
   A. Configure the encoder output channel on the servo axis drive to echo the encoder inputs of the H(horizontal) and V(vertical) axes.
   B. Issue this function one time for each servo axis drive. For more information about this function, see Drive Encode Output.

   The code snippets that follow show the correct syntax necessary to do this operation based on your configuration. Refer to the applicable AeroScript code snippet.

   Tip: If you copy this code snippet from the PDF file, some of the formatting might be different from what you see in this guide. To see the full IFOV example program, refer to Section 8.1. IFOV-Setup-Library Program.

- Use this AeroScript code snippet for configurations that have an auxiliary encoder setup.

```
// Step 1: Configures an output channel to echo encoder signals
// from the specified input channel.
// Applies to Auxiliary Encoder pins.
DriveEncoderOutputConfigureInput($HServo,
EncoderOutputChannel.AuxiliaryEncoder,
EncoderInputChannel.PrimaryEncoder)
DriveEncoderOutputConfigureInput($VServo,
EncoderOutputChannel.AuxiliaryEncoder,
EncoderInputChannel.PrimaryEncoder)
```

- Use this AeroScript code snippet for configurations that use SYNC ports.

```
// Step 1: Configures an output channel to echo encoder signals
// from the specified input channel.
// Applies to SYNC ports.
DriveEncoderOuputConfigureInput($HServo,
EncoderOutputChannel.SyncPortA,
EncoderInputChannel.PrimaryEncoder)
DriveEncoderOuputConfigureInput($VServo,
EncoderOutputChannel.SyncPortB,
EncoderInputChannel.PrimaryEncoder)
```

- Use this AeroScript code snippet for configurations that use an XR3 HSOUT connection.

```
// Step 1: Configures an output channel to echo encoder signals
// from the specified input channel.
// Applies to an XR3 HSOUT connection.
DriveEncoderOutputConfigureInput($HServo, EncoderOutputChannel.HSOUT,
EncoderInputChannel.PrimaryEncoder)
DriveEncoderOutputConfigureInput($VServo, EncoderOutputChannel.HSOUT,
EncoderInputChannel.PrimaryEncoder)
```

2. Issue the **`DriveEncoderOutputOn()`** function to turn on the encoder output channel of the servo axis. For more information about this function, see Controlling the Drive Encode Output. Do the steps that follow:
   A. For each servo axis, make sure this function uses the same encoder output channel that you configured in **Step 1**. If **`DriveEncoderOutputOn()`** uses a different channel, the encoder output will not operate correctly. For more information about this function, see Controlling the Drive Encoder Output.
   B. Issue this function one time for each servo axis drive. You must issue it after the **`DriveEncoderOutputConfigureInput()`** function. If **`DriveEncoderOutputOn()`** is issued before this function, the encoder output will not operate correctly. For more information about the **`DriveEncoderOutputConfigureInput()`** function, see Configuring the Drive Encoder Output.

The code snippets that follow show the correct syntax necessary to do this operation based on your configuration. Refer to the applicable AeroScript code snippet.

- Use this AeroScript code snippet for configurations that have an auxiliary encoder setup.

```
// Step 2: Echo the encoder signals from the H (horizontal)
// and V (vertical) servo drives.
// Applies to Auxiliary Encoder pins.
DriveEncoderOutputOn($HServo, EncoderOutputChannel.AuxiliaryEncoder,
EncoderOutputMode.Quadrature)
DriveEncoderOutputOn($VServo, EncoderOutputChannel.AuxiliaryEncoder,
EncoderOutputMode.Quadrature)
```

- Use this AeroScript code snippet for configurations that use the SYNC ports.

```
// Step 2: Echo the encoder signals from the H (horizontal)
// and V (vertical) servo drives.
// Applies to SYNC ports.
DriveEncoderOutputOn($HServo, EncoderOutputChannel.SyncPortA,
EncoderOutputMode.Quadrature)
DriveEncoderOutputOn($VServo, EncoderOutputChannel.SyncPortB,
EncoderOutputMode.Quadrature)
```

- Use this AeroScript code snippet for configurations that use an XR3 HSOUT connection.

```
// Step 2: Echo the encoder signals from the H (horizontal)
// and V (vertical) servo drives.
// Applies to an XR3 HSOUT connection.
DriveEncoderOutputOn($HServo, EncoderOutputChannel.HSOUT,
EncoderOutputMode.Quadrature)
DriveEncoderOutputOn($VServo, EncoderOutputChannel.HSOUT,
EncoderOutputMode.Quadrature)
```

After you issue these functions and complete the setup steps, the encoder outputs of the servo axes are turned on. At this time, you can read the galvo scanner drive signals as encoder signals on either the Auxiliary Encoder feedback or the **SYNC A** and **B** feedback channels. The specific feedback channel that is used is determined by the input you select for the scanner drive.

At this time, you can issue the `IfovOn()` function. After you do this, any motion code that you execute will be done with the servo and galvo scanner axes by commanding only the galvo scanner axes to move. After motion is completed, you must issue the `IfovOff()` function, which causes any commands sent to the galvo scanner to be only within its specified field of view (FOV). Thus, no servo motion will be executed. For more information about these functions, see IFOV Functions.

Chapter 5: IFOV System Verification

After you configure your IFOV system and make an AeroScript program with the functions from the previous sections, you must verify IFOV functionality before your run any programs. In Automation1 Status Utility, use the **Diagnostics** tab to make sure that all the axis encoder signals are configured and that the controller is tracking the signals correctly. For more information, see Automation1 Status Utility Overview.

You must monitor the signals that follow:

- **Position Feedback** of the galvo scanner and servo axes:
  - On the servo axes (**SX** and **SY**), this signal shows the multiplied position counts from servo stages.
  - On the galvo scanner axes (**GX** and **GY**), this signal shows the multiplied position counts from the galvo scanner stages.
- **Auxiliary Feedback** of the galvo scanner axes. This applies only if you use an Auxiliary Encoder Output with your drive or an Automation1 XR3 with High Speed Output (HSOUT) to output the encoder signals:
  - On the servo axes (**SX** and **SY**), **Auxiliary Feedback** signals do not show counts or data.
  - For the **Auxiliary Feedback** signals of the galvo scanner axes (**GX** and **GY**), the related collinear servo axis feedback scaled to galvo scanner counts per unit is shown when you scale it correctly.
- **SYNC Port A** and **SYNC Port B Position Feedback** of the galvo scanner axes. This applies only if you use SYNC ports to output the encoder signals:
  - For the **SYNC A** and **B** feedback of each servo axis, the output signal for the related SYNC port used on the servo axis shows a count value that matches the counts of the **Position Feedback** signal for that axis.
  - For the **Auxiliary Feedback** signals of the galvo scanner axes (**GX** and **GY**), the related collinear servo axis feedback scaled to galvo scanner counts per unit is shown when you scale it correctly.

How to verify IFOV functionality

IMPORTANT: Do these steps in the order that they are shown.

1. Make sure the axes are enabled, homed, and counts are tracking on all the **Position Feedback** outputs of the galvo scanner axes and servo axes. Look at the **Position Feedback** output on all the galvo scanner axes and servo axes to make sure some jitter occurs. Also make sure that the axes are jogging and corresponding counts are shown in the output.
2. Make sure the encoder outputs are configured correctly and are turned on. The Drive Encoder Output must be configured and turned on as discussed in Section 4.2.6. Configure and Echo Servo Encoder Signals to Galvo Scanner Axes. If you do not issue the correct functions, this will cause the servo axis feedback to not be included in readouts for galvo scanner feedback that uses an auxiliary encoder setup or SYNC ports. As a result, the galvo scanner axis cannot track the servo axis position when IFOV is enabled.

   Tip: To verify encoder output tracking, refer to the procedure that follows.

   #### How to verify encoder output tracking

   A. Insert a debug breakpoint into your program. Put it after the IFOV setup code but before the `IfovOn()` function. This breakpoint will pause program execution after the encoder outputs are enabled but before IFOV is turned on.

   B. Make sure the encoder outputs are enabled and scaled correctly.

   C. Jog the servo axes (**SX** and **SY**) manually.

   D. Monitor the galvo scanner axes (**GX** and **GY**) in the Status Utility. Examine the feedback values that follow:

   - **Auxiliary Feedback**
   - **SYNC Port A Position**
   - **SYNC Port B Position**

3. Make sure the encoder output from the servo axes is scaled correctly on the auxiliary feedback from the galvo scanner axes. Section 4.2.5. Configure and Define IFOV Scale Factors and Axis Pairs shows you how to set the IFOV axis pairs. The procedure that follows will show you how to make sure they are set up correctly. In Automation1 Studio, do the steps that follow with the encoder outputs turned on and the `IfovSetAxisPairs()` function correctly defined:

How to verify the IFOV axis pairs setup

A. Jog each servo axis (**SX** and **SY**) to a specified distance, such as 1 mm.

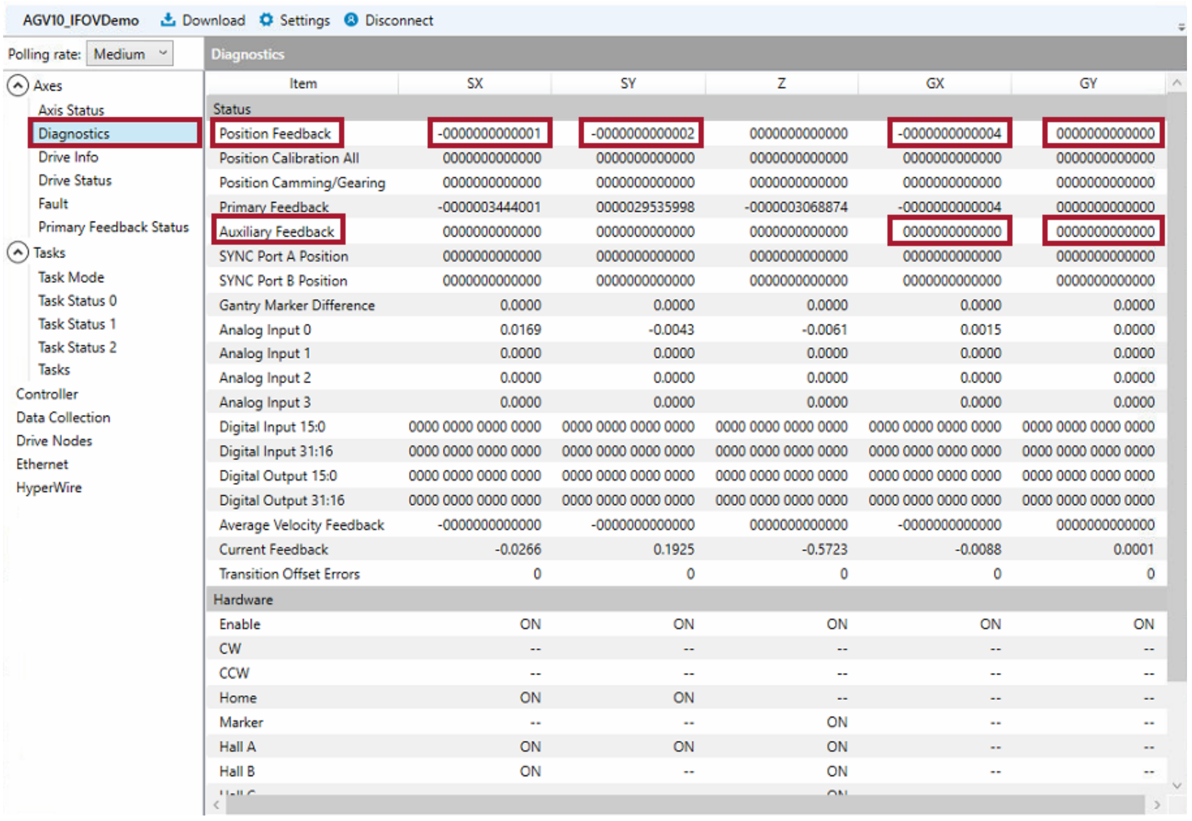
B. Jog each galvo scanner axis (**GX** and **GY**) to the same distance that you specified in **Step A**.

C. Compare the **Position Feedback** value from each galvo scanner axis to its corresponding **Auxiliary Feedback**, **SYNC Port A Position**, or **SYNC Port B Position** as reported in the Status Utility. These values should be approximately the same.

D. While you do this, verify the items that follow as reported in the Status Utility:

- The **Auxiliary Feedback**, **SYNC Port A Position**, or **SYNC Port B Position** changes on the galvo scanner axes (**GX** and **GY**).
- The magnitude and direction of the **Auxiliary Feedback** count change match the **Position Feedback** count change on the galvo scanner axis.

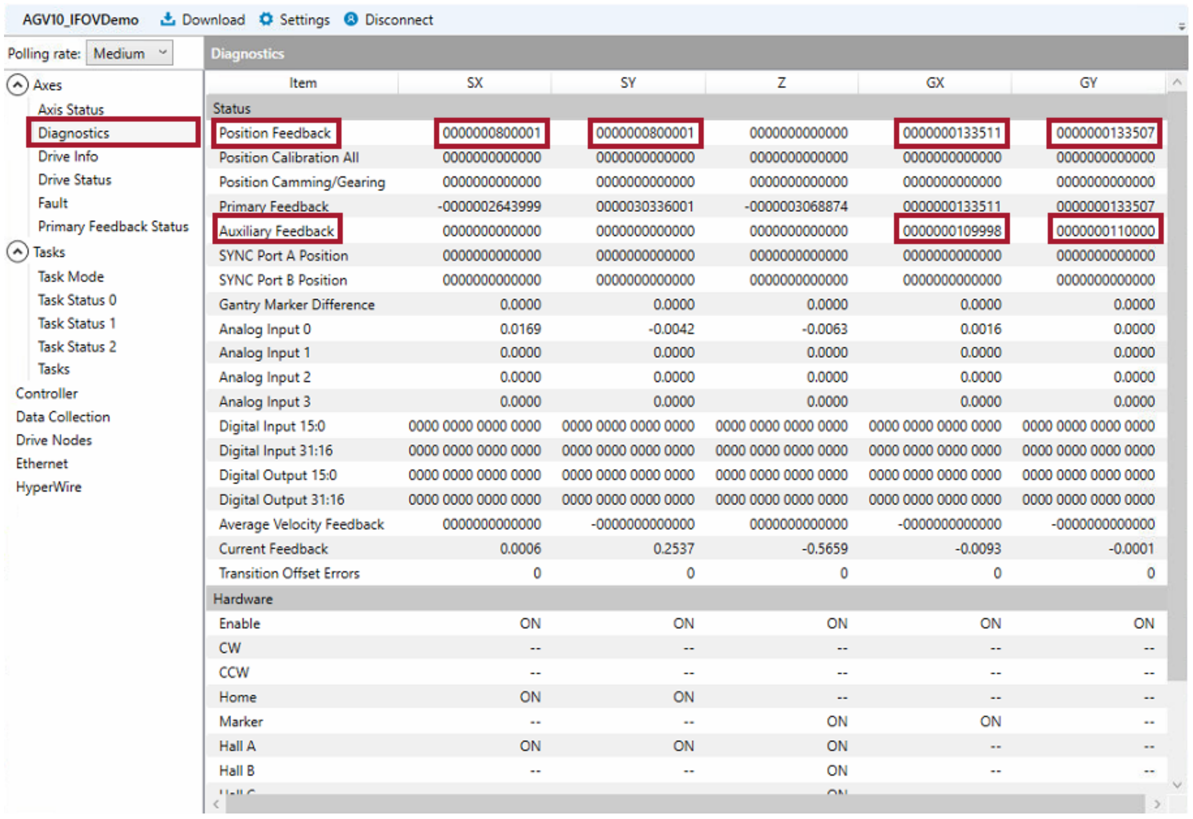
Figure 5-1 that follows shows the Status Utility before the 1 mm move.



Item	SX	SY	Z	GX	GY
Status					
Position Feedback	-0000000000001	-0000000000002	0000000000000	-0000000000004	0000000000000
Position Calibration All	0000000000000	0000000000000	0000000000000	0000000000000	0000000000000
Position Camming/Gearing	0000000000000	0000000000000	0000000000000	0000000000000	0000000000000
Primary Feedback	-0000003444001	0000029535998	-0000003068874	-0000000000004	0000000000000
Auxiliary Feedback	0000000000000	0000000000000	0000000000000	0000000000000	0000000000000
SYNC Port A Position	0000000000000	0000000000000	0000000000000	0000000000000	0000000000000
SYNC Port B Position	0000000000000	0000000000000	0000000000000	0000000000000	0000000000000
Gantry Marker Difference	0.0000	0.0000	0.0000	0.0000	0.0000
Analog Input 0	0.0169	-0.0043	-0.0061	0.0015	0.0000
Analog Input 1	0.0000	0.0000	0.0000	0.0000	0.0000
Analog Input 2	0.0000	0.0000	0.0000	0.0000	0.0000
Analog Input 3	0.0000	0.0000	0.0000	0.0000	0.0000
Digital Input 15:0	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
Digital Input 31:16	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
Digital Output 15:0	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
Digital Output 31:16	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
Average Velocity Feedback	-0000000000000	-0000000000000	0000000000000	-0000000000000	0000000000000
Current Feedback	-0.0266	0.1925	-0.5723	-0.0088	0.0001
Transition Offset Errors	0	0	0	0	0
Hardware					
Enable	ON	ON	ON	ON	ON
CW	--	--	--	--	--
CCW	--	--	--	--	--
Home	ON	ON	--	--	--
Marker	--	--	ON	--	--
Hall A	ON	ON	ON	--	--
Hall B	ON	--	ON	--	--

Figure 5-1: Status Utility Before the Move

Figure 5-2 that follows shows the Status Utility after the 1 mm move.



Item	SX	SY	Z	GX	GY
Status					
Position Feedback	0000000800001	0000000800001	0000000000000	0000000133511	0000000133507
Position Calibration All	0000000000000	0000000000000	0000000000000	0000000000000	0000000000000
Position Camming/Gearing	0000000000000	0000000000000	0000000000000	0000000000000	0000000000000
Primary Feedback	-0000002643999	0000030336001	-0000003068874	0000000133511	0000000133507
Auxiliary Feedback	0000000000000	0000000000000	0000000000000	0000000109998	0000000110000
SYNC Port A Position	0000000000000	0000000000000	0000000000000	0000000000000	0000000000000
SYNC Port B Position	0000000000000	0000000000000	0000000000000	0000000000000	0000000000000
Gantry Marker Difference	0.0000	0.0000	0.0000	0.0000	0.0000
Analog Input 0	0.0169	-0.0042	-0.0063	0.0016	0.0000
Analog Input 1	0.0000	0.0000	0.0000	0.0000	0.0000
Analog Input 2	0.0000	0.0000	0.0000	0.0000	0.0000
Analog Input 3	0.0000	0.0000	0.0000	0.0000	0.0000
Digital Input 15:0	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
Digital Input 31:16	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
Digital Output 15:0	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
Digital Output 31:16	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
Average Velocity Feedback	0000000000000	-0000000000000	0000000000000	-0000000000000	-0000000000000
Current Feedback	0.0006	0.2537	-0.5659	-0.0093	-0.0001
Transition Offset Errors	0	0	0	0	0
Hardware					
Enable	ON	ON	ON	ON	ON
CW	--	--	--	--	--
CCW	--	--	--	--	--
Home	ON	ON	--	--	--
Marker	--	--	ON	ON	--
Hall A	ON	ON	ON	--	--
Hall B	--	--	ON	--	--

Figure 5-2: Status Utility After the Move

Chapter 6: IFOV System Calibration

You must have a metrology tool in order to do the calibration procedure that follows. Examples of this tool include a calibrated camera or a microscope with micrometer stages. Use the most accurate measurement method available to correctly measure the travel of the galvo scanner axes (**GX** and **GY**).

Tip: Aerotech recommends that you physically mark or cut into a surface with a laser that has the correct input beam diameter and wavelength.

IMPORTANT: You can do a check of the visual alignment by using a laser pointer to project a spot onto an accurate scale pattern. But when multiple operators use this visual method for the same check, different results can occur. The difference in these results causes measurement interpretation error.

6.1. Verify the Scaling of the Galvo Scanner Axes

During the Machine Setup process, a lens is assigned to the galvo scanner axes that requires you to enter these lens-specific values: **Effective Focal Length (EFL)** and **Field of View (FOV)**. By entering these values, the calculation is set for the galvo scanner axes (**GX** and **GY**) CountsPerUnit. This specifies the number of encoder counts per user defined units, such as per millimeter. As a result, you must install the galvo scanner with the lens spaced at the correct distance from the mirrors (immersion depth). Then you must space the assembly in **Z** from the workpiece at the specified working distance of the lens from the lens manufacturer data sheet.

To verify that the motion of the laser on the surface is scaled correctly, do the steps that follow:

How to verify scaling of laser motion on the surface

1. In Automation1 Studio, home the galvo scanner axes (**GX** and **GY**) to (0,0).
2. Mark a simple crosshair (+). Make it symmetric about the home position with a length of exactly 1 millimeter.
3. Use your metrology tool (or most accurate measurement method available) to measure the length of each line. Each line must be equal to 1.000 mm ∓0.001 mm.

   IMPORTANT: The accuracy and repeatability of the metrology tool (or most accurate measurement method available) will have an effect on the accuracy of the scaling setup.

4. Based on the results of your measurement, do one of the options that follow:
   - If the lines are equal to the specified length in the two directions, the setup is accurate. Thus, the procedure is complete.
   - If the lines are not equal to the specified length, do the procedure that follows to verify the scaling setup. Then try to verify the scaling of laser motion on the surface again.

   #### How to verify the scaling setup

   A. Make sure that the lens mount design and lens spacing (M2 Distance) are correct. You can find these dimensions in the AGV galvo scanner hardware manuals. To download your manual from www.aerotech.com, go to the **Mechanical** section of Manuals & Help Files.

   B. Make sure that the working distance of the lens is accurate.

   C. Make sure that the lens EFL is specified correctly in Machine Setup.

   D. Make sure that the parallelism of lens to AGV and the parallelism of AGV to workpiece are correct.

If you cannot verify this setup, incorrectly scaled motion might be performed by the galvo scanner during processing.

6.2. Correcting Galvo Scanner and Servo Rotational Misalignment

For IFOV to operate correctly, you must correct any mechanical misalignment that occurs between the galvo scanner axes and servo axes. Use the `GalvoRotationSet()` function to rotate the galvo scanner axis pair (**GX** and **GY**) relative to the servo axis pair (**SX** and **SY**). This function specifies a value, in degrees, for the relative rotation between the galvo scanner and servo axis pairs. Each of these pairs are usually orthogonal pairs, which give you a singular rotation value. For more information about this function, see Galvo Functions. To measure a value for this function, refer to the procedure that follows.

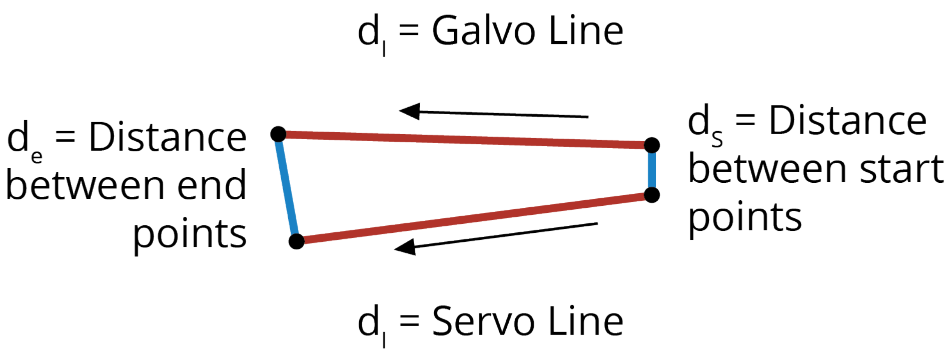

d_l = Galvo Line
d_e = Distance between end points
d_s = Distance between start points
d_l = Servo Line

Figure 6-1: Galvo and Servo Line Measurements

How to measure the GalvoRotationSet() value

1. Mark a line that uses only one galvo scanner axis (**GX** or **GY**). To get the most accurate measurements, mark the longest line that is available for the galvo scanner field of view (FOV) or stage travel.
2. Return the galvo scanner to its zero location.
3. Mark a line of the same length from the start point to the same end point. Use only the collinear servo axis (**SX** for **GX**) or (**SY** for **GY**).
4. For the servo axis that is not being used, jog it to a specified distance (d_s). Then record this value. This prevents confusion about where the start of each line is.
5. Measure the length of each marked line and make sure they are the same. If they are not, the servo and galvo scanner axes are not scaled correctly.
6. Measure the distance between the end points of the two lines (d_e).
7. Calculate the relative angle between the galvo scanner and servo axes. Use the equation that follows:

$$\text{Angle}\,\theta = \text{asin}\left(\frac{d_e - d_s}{\text{length of line}}\right)$$

Equation: 6-1: Relative Angle

If the coordinate systems are accurately aligned, the delta between the start and end points separation is zero. If the coordinate systems are not aligned ($d_e \neq d_s$), you must use the `GalvoRotationSet()` function to rotate the galvo scanner coordinate system relative to the servo axes. Issue this function within your program. Then do the How to measure the GalvoRotationSet() value procedure again and make sure the separation between the start and end points is zero ($d_e = d_s$). For more information about this function, see GalvoRotationSet() function.

6.3. Calibrating Galvo Scanner Lens FOV

To compensate for lens errors, make sure that you calibrate the galvo scanner lens installed on the IFOV system. If you do not calibrate the lens or you calibrate it incorrectly, this will cause incorrect laser positioning at different locations within the galvo scanner field of view (FOV). It will also cause incorrect servo axis commands to get geometrically accurate motion when IFOV motion is being executed.

To calibrate the 2D galvo scanner, either load an existing calibration file (`.cal`) or make one by using the Calibration Module in the **Configure** workspace of Automation1 Studio.

For information about how to format a galvo scanner calibration file (`.cal`), see Galvo 2D Axis Calibration File Format.

You can also dynamically load calibration files through the AeroScript functions discussed in Calibration Functions.

After you load a calibration file, you must do the mark and measure procedure to make sure that each mark location is accurate. For instructions about how to do this, see the Interpolation section of Calibration Topic.

Chapter 7: Position Synchronized Output (PSO) with IFOV

Position Synchronized Output (PSO) and Part-Speed PSO are compatible with IFOV configurations. The procedure to configure and use PSO with an IFOV configuration is almost the same as for a standard Aerotech drive. But you must do the additional steps that follow. For an example of PSO functionality, see PSO Fixed Distance Pulse Output (GL4) Example Program.

You must make sure that the PSO event trigger accurately tracks the scanner and stages together. To do this, refer to the procedure that follows:

How to configure the PSO event trigger

1. Refer back to **Step 1** of the How to set the IFOV axis pairs procedure in this guide.
2. Find the scale factor that you calculated for each collinear axis pair.
3. For each galvo axis (**GX** and **GY**), set the PrimaryEmulatedQuadratureDivider Parameter to the IFOV scale factor that corresponds to its collinear axis pair. Refer to the code examples that follow.

Issue the commands that follow from within a user program immediately after the PSO setup commands. These commands will make sure that the PrimaryEmulatedQuadratureDivider parameters are correctly set based on the specific IFOV configuration that you are using.

Tip: If you copy this code snippet from the PDF file, some of the formatting might be different from what you see in this guide. To see the full IFOV example program, refer to Section 8.1. IFOV-Setup-Library Program.

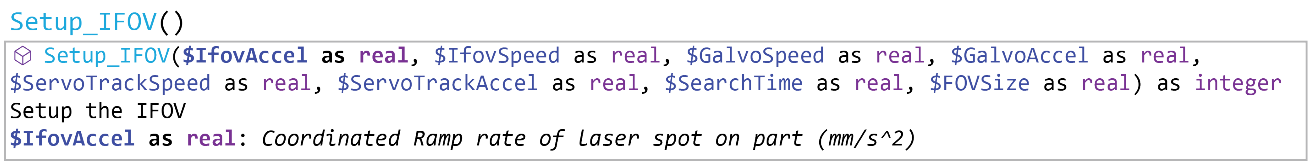
```
// Set the PrimaryEmulatedQuadratureDivider parameter of the first galvo axis
// equal to the IFOV scale factor for the first galvo/servo axis pair.
ParameterSetAxisValue($HGalvo, AxisParameter.PrimaryEmulatedQuadratureDivider,
(ParameterGetAxisValue($HGalvo,
AxisParameter.CountsPerUnit)/ParameterGetAxisValue
($HServo, AxisParameter.CountsPerUnit))*ParameterGetAxisValue($HServo,
AxisParameter.PrimaryEmulatedQuadratureDivider)
```

```
// Set the PrimaryEmulatedQuadratureDivider parameter of the second galvo axis
// equal to the IFOV scale factor for the second galvo/servo axis pair.
ParameterSetAxisValue($VGalvo, AxisParameter.PrimaryEmulatedQuadratureDivider,
(ParameterGetAxisValue($VGalvo,
AxisParameter.CountsPerUnit)/ParameterGetAxisValue
($VServo, AxisParameter.CountsPerUnit))*ParameterGetAxisValue($VServo,
AxisParameter.PrimaryEmulatedQuadratureDivider)
```

Chapter 8: Write Your IFOV Program

This section helps you make AeroScript programs that use the Infinite Field of View (IFOV) feature. IFOV makes the programming process easier because your AeroScript program commands motion only to the galvo scanner axes after it is configured and enabled with the **`IfovOn()`** function. Then the IFOV algorithm of the Automation1 controller automatically calculates and executes the necessary motion for the servo axes to make sure the laser spot follows the commanded trajectory. At the same time, it also optimizes the servo stage position to keep the laser spot biased toward the center of the defined field of view (FOV) of the galvo scanner. This optimization keeps optical distortion to a minimum, which is caused by the F-Theta lens when the scanner FOV moves.

The example that follows shows an AeroScript library that you can configure in the Program Automation Module as a compiled library file. When you use this method to configure the controller, you can issue the `Setup_IFOV()` function from within any task or program. Then you can define all the attributes of the IFOV configuration from one line of code, as shown in Figure 8-1 that follows.



```
Setup_IFOV()
Setup_IFOV($IfovAccel as real, $IfovSpeed as real, $GalvoSpeed as real, $GalvoAccel as real,
$ServoTrackSpeed as real, $ServoTrackAccel as real, $SearchTime as real, $FOVSize as real) as integer
Setup the IFOV
$IfovAccel as real: Coordinated Ramp rate of laser spot on part (mm/s^2)
```

Figure 8-1: Argument Definitions for the `Setup_IFOV()` Function

Figure 8-1 shows the `Setup_IFOV()` function, which requires you to specify inputs for the arguments that follow:

Arguments

- *`$IfovAccel`* - Coordinated ramp rate of the laser spot on the part (mm/s^2).
- *`$IfovSpeed`* - Coordinated speed of the laser spot on the part (mm/s).
- *`$GalvoSpeed`* - Axis Coordinated Speed for the galvo axes during jump moves (mm/s). To use the maximum speed permitted by IFOV, set this argument to 0. For a different speed, set this argument to a specific value.
- *`$GalvoAccel`* - Axis Coordinated Acceleration for the galvo axes during jump moves (mm/s^2).
- *`$FOVSize`* - The field of view size (mm x mm).
- *`$ServoTrackSpeed`* - The maximum servo axis speed (mm/s).
- *`$ServoTrackAccel`* - The maximum servo axis tracking acceleration (mm/s^2).
- *`$SearchTime`* - The maximum search time the controller looks ahead in IFOV (ms). [Start with 200 ms.]

You can specify the `Setup_IFOV()` function as necessary within a single program. Before you specify argument values, you must use the **`IfovOff()`** function to turn off IFOV. Then you can specify your values. When that is complete, use the **`IfovOn()`** function to turn on IFOV again.

When IFOV is enabled through the `IfovOn()` function (as shown in the example program that follows), motion commands such as `G1` or `MoveLinear()` are directed toward the galvo scanner axes (e.g., **GX**, **GY**, or $HGalvo, $VGalvo). For more information about these functions, see Linear Motion.

The advanced IFOV algorithm integrated within the Automation1 controller manages the servo stage motion for you. It continuously calculates the required trajectory for the servo axes to make sure the combined movement of the servo and galvo axes results in the laser spot accurately following the path commanded to the galvo axes. The algorithm keeps the laser spot near the center of the defined field of view (FOV) by commanding the servo stages to move as necessary. Thus, you can focus on your specified laser spot path on the workpiece surface, while the controller handles the complex, synchronized movements of the fast galvo motors and the slower, larger-travel servo stages.

8.1. IFOV-Setup-Library Program

For instructions about how to download and use this program, refer to the procedure that follows.

How to use the IFOV-Setup-Library Program

1. Click IFOV-Setup-Library.ascriptlib to download the IFOV Library setup program.
2. Open Automation1 Studio. Then select the **Develop** workspace.
3. On the **Programming** toolbar, click the **Open File** button. Then go to the folder where you downloaded the program. This is usually your **Downloads** folder.
4. Open the **IFOV-Setup-Library** program in Automation1 Studio.
5. Based on the information in this guide, make any necessary changes to this program for your IFOV project.
6. On the **Programming** toolbar, click the arrow adjacent to the **Save** button. Then select **Save As**.
7. Save the file as an `.ascriptlib` file. Aerotech recommends that you save it to your **Automation1** folder.
8. Click the **Build** button to compile the program.
9. Load the `.a1lib` file into the **Program Automation** module. See the How to get your compiled AeroScript library files procedure for more information.

IMPORTANT: The AeroScript example program that follows spans multiple pages. It shows you how this program will look in Automation1 Studio. For information about how to download and use this program, refer to How to use the IFOV-Setup-Library Program.

```
/*****************************************************************************
*****
Title:
IFOV_Setup

Description:
This library will configure the IFOV. It will calculate all
the necessary parameters and set their values.

Functions:
Setup_IFOV

Changes:
11-01-2025 - Initial Revision 1.0

*****************************************************************************
*** */

/*!
    @summary Setup the IFOV.
    @argument $IfovAccel Coordinated ramp rate of the laser spot on
    the part (mm/s^2).
    @argument $IfovSpeed Coordinated speed of the laser spot
    on the part (mm/s).
    @argument $GalvoSpeed Axis Coordinated Speed for the galvo axes
    during jump moves (mm/s). To use the maximum speed permitted by
    IFOV, set this argument to 0. For a different speed,
    set this argument to a specific value.
    @argument $GalvoAccel Axis Coordinated Acceleration for
    the galvo axes during jump moves (mm/s^2).
    @argument $FOVSize The field of view size (mm x mm).
    @argument $ServoTrackSpeed The maximum servo axis speed (mm/s).
    @argument $ServoTrackAccel The maximum servo axis tracking
    acceleration (mm/s^2).
    @argument $SearchTime The maximum search time the controller
    looks ahead in IFOV (ms). [Start with 200 ms]
    @return Response
*/
library function Setup_IFOV($IfovAccel as real, $IfovSpeed as real,
$GalvoSpeed as real, $GalvoAccel as real, $ServoTrackSpeed as real,
$ServoTrackAccel as real, $SearchTime as real, $FOVSize as real) as integer
```

```
//==============================================================================
//======================================

// Define Axes
var $HGalvo as axis = GX
var $VGalvo as axis = GY
var $HServo as axis = SX
var $VServo as axis = SY

// Define the IFOV axes. $axisPairH is the horizontal pair (Servo & Galvo).
// $axisPairV is the vertical pair (Servo & Galvo).
// Define the collinear H Pair for the IFOV setup.
var $axisPairH[] as axis = [$HGalvo, $HServo]
// Define the collinear V Pair for the IFOV setup.
var $axisPairV[] as axis = [$VGalvo, $VServo]
// Define all the IFOV axes. (This does not include the Z axis.)
var $allAxes[] as axis = [$HGalvo, $HServo, $VGalvo, $VServo]

// Disable IFOV.
IfovOff()

    // Make sure that IFOV is disabled on all the IFOV axes.
    foreach var $axis in $allAxes
        wait (StatusGetAxisItem($axis,
        AxisStatusItem.AxisStatus, AxisStatus.IfovEnabled)
        != AxisStatus.IfovEnabled)
    end

//==============================================================================
//======================================

// Define and configure the maximum permitted galvo speed.
// Time Granularity (must be 1 or 5) is a value of time in ms that
// defines the minimum update rate for the IFOV algorithm to do
// its operations. The default value is 5 ms and
// the minimum value is 1 ms.
var $IfovTimeGranularity as integer = 1
// Set $IfovTimeGranularity to the specified value.
IfovSetup(IfovSetupItem.TimeGranularity, $IfovTimeGranularity)

// Variable that sets the maximum galvo speed
// that is permitted during IFOV operations.
var $GalvoSpeedSet as real

    if ($GalvoSpeed == 0)
        $GalvoSpeedSet = (500*$FOVSize)/$IfovTimeGranularity
    else
        $GalvoSpeedSet = $GalvoSpeed
    end

// Set IFOV commands for the maximum speeds, accelerations,
// FOV size, search time, and tracking.
// Configures the Coordinated Ramp Type for how the
// laser will move on the part surface.
SetupCoordinatedRampType(RampType.Sine)
// Configures the Coordinated Ramp Rate for the
// laser spot on the part surface.
SetupCoordinatedRampValue(RampMode.Rate, $IfovAccel)
// Configures the Coordinated Speed for the
// laser spot on the part surface.
SetupCoordiantedSpeed($IfovSpeed)
// ***When IFOV is turned on, you must specify the
// galvo ramp and speeds. IFOV will ignore the
// servo axis ramp and speeds.
// Specify the ramp type for the galvo axes.
SetupAxisRampType([$HGalvo, $VGalvo], RampType.Sine)
// Specify the target ramp rate for the galvo axes.
SetupAxisRampValue([$HGalvo, $VGalvo], RampMode.Rate, $GalvoAccel)
// Specify the target speed for the galvo axes.
SetupAxisSpeed([$HGalvo, $VGalvo], [$GalvoSpeedSet, $GalvoSpeedSet])
// Configures the maximum search time in ms. Start with 200.
IfovSetTime($SearchTime)
// Configures the maximum speed of the servo axes to
// track during IFOV motion.
IfovSetTrackingSpeed($ServoTrackSpeed)
// Configure the maximum ramp rate of the
// servo axes to track during IFOV motion.
IfovSetTrackingAcceleration($ServoTrackAccel)
// An empty axis list is passed, which means
// that no additional axes are synchronized with IFOV motion.
// If you want to synchronize more axes
// with IFOV motion, pass them to the function.
IfovSetSyncAxes([])
```

```
// Configures the IFOV field of view (FOV) size.
IfovSetSize($FOVSize)
// The maximum data rate output for all
// the Automation1 servo drives (25 MHz).
// To include jitter, set this value to 22 MHz.
var $MaxDataRate as integer = 22000000

//==============================================================================
//======================================

// The equations that follow calculate the
// Maximum Data Rate (counts/s) for multiplied encoder signals
// at the servo drive encoder output. This is done
// to calculate the correct divider parameter to make
// sure the hardware limited rate at 22 MHz is not exceeded.
    var $HServoMaxDat as real = $ServoTrackSpeed *
ParameterGetAxisValue($HServo, AxisParameter.CountsPerUnit)
    var $VServoMaxDat as real = $ServoTrackSpeed *
ParameterGetAxisValue($VServo, AxisParameter.CountsPerUnit)
    Dwell(0.1)
// Calculates the required EmulatedQuadratureDivider for
// $HServo based on the maximum data rate at
// the servo tracking speed that you specified.
    var $HDivider as real = Abs($HServoMaxDat/ $MaxDataRate)
// Calculates the required EmulatedQuadratureDivider for
// $VServo based on the maximum data rate at
// the servo tracking speed that you specified.
    var $VDivider as real = Abs($VServoMaxDat/ $MaxDataRate)
    Dwell(0.1)

// The if statements that follow do a check to make sure
// that the value of the PrimaryEmulatedQuadratureDivider parameter
// obeys specific requirements before you set it.
    // Cannot be less than 1.
    // If the value is less, you must clamp it to 1.
    // This is a requirement.
    if $HDivider < 1
        // If < 1, set PrimaryEmulatedQuadratureDivider == 1
        ParameterSetAxisValue($HServo, AxisParameter.
        PrimaryEmulatedQuadratureDivider, 1)

    else
```

```
            // If > 1, do a check to see if the
            // (PrimaryEncoderMultiplicationFactor /
            // PrimaryEmulatedQuadratureDivider)
            // is an integer.
            var $HSearch as integer = Trunc(Ceil($HDivider))
            var $HMultiFactor as integer = Trunc(ParameterGetAxisValue($HServo,
            AxisParameter.PrimaryEncoderMultiplicationFactor))
            while (($HMultiFactor / $HSearch * $HSearch != $HMultiFactor)
                    $HSearch++
                    if ($HSearch >= $HMultiFactor)
                        $HSearch = $HMultiFactor
                        break
                    end
            end
            ParameterSetAxisValue($HServo, AxisParameter.
            PrimaryEmulatedQuadratureDivider, $HSearch)

            end

    Dwell(0.01)

        // Cannot be less than 1.
        // If the value is less, you must clamp it to 1.
        // This is a requirement.
        if $VDivider < 1
            // If < 1, set PrimaryEmulatedQuadratureDivider == 1
            ParameterSetAxisValue($VServo, AxisParameter.
            PrimaryEmulatedQuadratureDivider, 1)
        else
            // If > 1, do a check to see if the
            // (PrimaryEncoderMultiplicationFactor /
            // PrimaryEmulatedQuadratureDivider)
            // is an integer.
            var $VSearch as integer = Trunc(Ceil($VDivider))
            var $VMultiFactor as integer =
            Trunc(ParameterGetAxisValue($VServo,
            AxisParameter.PrimaryEncoderMultiplicationFactor))
            while (($VMultiFactor / $VSearch) * $VSearch != $VMultiFactor)
                $VSearch++
                if($VSearch >= $VMultiFactor)
                    $VSearch = $VMultiFactor
                    break
                end
            end
            ParameterSetAxisValue($VServo,
            AxisParameter.PrimaryEmulatedQuadratureDivider, $VSearch)

        end
        Dwell(0.01)

    // Define and calculate encoder scale factors for
    // $scaleFactorH($HGalvo, $HServo) and $scaleFactorV($VGalvo, $VServo).
    var $scaleFactorH as real = (ParameterGetAxisValue
    ($HGalvo, AxisParameter.CountsPerUnit)/ParameterGetAxisValue
    ($HServo, AxisParameter.CountsPerUnit))*ParameterGetAxisValue
    ($HServo, AxisParameter.PrimaryEmulatedQuadratureDivider)
    var $scaleFactorV as real = (ParameterGetAxisValue
    ($VGalvo, AxisParameter.CountsPerUnit)/ParameterGetAxisValue
    ($VServo, AxisParameter.CountsPerUnit))*ParameterGetAxisValue
    ($VServo, AxisParameter.PrimaryEmulatedQuadratureDivider)
    Dwell(0.01)
    // Define the IFOV axis pairs with the
    // calculated IFOV encoder scale factors.
    IfovSetAxisPairs($axisPairH, $axisPairV, $scaleFactorH, $scaleFactorV)
    //============================================================================
    //======================================
    // Initialize the stages and IFOV.
    // Before you turn on IFOV, make sure that
    // all the galvo and servo axes
    // are enabled and homed.
    // Make sure that all the IFOV axes are enabled.
    var $allAxesEnabled as integer = 1
    var $isEnabled as integer = 0

        foreach var $axis in $allAxes
            $isEnabled = (StatusGetAxisItem($axis,
            AxisStatusItem.DriveStatus, DriveStatus.
            Enabled) == DriveStatus.Enabled)
            $allAxesEnabled = $allAxesEnabled & $isEnabled
        end

        // Enable the axes only if they are
        // not currently enabled.
        if $allAxesEnabled != 1
            Enable($allAxes)
        end

    // Make sure that all the IFOV axes are homed.
    var $allAxesHomed as integer = 1
    var $isHomed as integer = 0

        foreach var $axis in $allAxes
            $isHomed = (StatusGetAxisItem($axis,
            AxisStatusItem.AxisStatus, AxisStatus.
            .Homed) == AxisStatus.Homed)
            $allAxesHomed = $allAxesHomed & $isHomed
        end

        // Home the axes only if they are not currently homed.
        if $allAxesHomed != 1
            Home($allAxes)
        end
    //=============================================================================
    //======================================

        // Use the statements that follow to configure the
        // servo encoder signal to output from the drive and
        // then to turn on the encoder output.
        // Step 1: Configure the output channel to echo encoder
        // signals from the input channel for each servo axis.
        // For program lines that apply to the auxiliary encoder or
        // SYNC cable wiring that is specific to your system,
        // keep them uncommented. For program lines that do not
        // apply, keep them commented. To help find the
        // applicable program lines, examine the
        // polarity and port labels.
        // Step 2: Echo the encoder signals from
        // the H and V servo drives. For program lines that apply to
        // the auxiliary encoder or to SYNC cable wiring that is
        // specific to your system, keep them uncommented.
        // For program lines that do not apply,
        // keep them commented. To help find the
        // applicable program lines, examine the
        // polarity and port labels. Also, find the program line with
        // the encoder output channel that corresponds to
        // your Machine Setup configuration. Keep this
        // line uncommented. Comment the program lines
        // with encoder output channels that do not apply.

    DriveEncoderOutputConfigureInput($HServo, EncoderOutputChannel.
    AuxiliaryEncoder, EncoderInputChannel.PrimaryEncoder)
    DriveEncoderOutputConfigureInput($VServo, EncoderOutputChannel.
    AuxiliaryEncoder, EncoderInputChannel.PrimaryEncoder)
    DriveEncoderOutputOn($HServo, EncoderOutputChannel.
    AuxiliaryEncoder, EncoderOutputMode.Quadrature)
    DriveEncoderOutputOn($VServo, EncoderOutputChannel.
    AuxiliaryEncoder, EncoderOutputMode.Quadrature)
    /*

    DriveEncoderOutputConfigureInput($HServo, EncoderOutputChannel.
    SyncPortA, EncoderInputChannel.PrimaryEncoder)
    DriveEncoderOutputConfigureInput($VServo, EncoderOutputChannel.
    SyncPortA, EncoderInputChannel.PrimaryEncoder)
    DriveEncoderOutputOn($HServo, EncoderOutputChannel.SyncPortA,
    EncoderOutputMode.Quadrature)
    DriveEncoderOutputOn($VServo, EncoderOutputChannel.SyncPortA,
    EncoderOutputMode.Quadrature)

    DriveEncoderOutputConfigureInput($HServo, EncoderOutputChannel.
    HighSpeedOutputs, EncoderInputChannel.PrimaryEncoder)
    DriveEncoderOutputConfigureInput($VServo, EncoderOutputChannel.
    HighSpeedOutputs, EncoderInputChannel.PrimaryEncoder)
    DriveEncoderOutputOn($HServo, EncoderOutputChannel.
    HighSpeedOutputs, EncoderOutputMode.Quadrature)
    DriveEncoderOutputOn($VServo, EncoderOutputChannel.
    HighSpeedOutputs, EncoderOutputMode.Quadrature)
    */
    return 1

    end
```

Glossary of IFOV Concepts

C

Collinear Axis Pair

A pair of axes that consists of one galvo scanner axis (GX) and one linear servo axis (X) that are parallel or collinear with each other.

E

Effective Focal Length (EFL)

The distance from the galvo scanner F-theta len's principal plane to the point where the laser beam is focused for a zero scan angle.

F

F-Theta Lens

A type of lens or optical objective usually used with galvo scanners to linearize the mirror rotation position to linear user units along the machine coordinate X/Y axes.

F-Theta Lens Working Distance

The working distance of an F-theta lens for laser processing is the distance from the front of the lens (or lens housing) to the point where the laser is focused on the material.

Field of View (FOV)

The travel limits of where the laser spot can move for a specified galvo scanner setup. This will define the software and end-of-travel limits for each galvo scanner axis.

G

Galvo Scanner

A galvo scanner (short for galvanometer scanner), is a multi-axis (usually 2-axis) mirror positioning device. It contains optical elements in the form of mirrors that are designed to intake a laser beam at a specified diameter and position it onto a working surface. This is done by moving an orthogonal set of mirrors, which translates the beam onto a special optical objective known as an F-Theta lens. As a result, you can program and execute a trajectory in a 2D plane in Cartesian (X-Y) coordinates.

Galvo Scanner Axis

A galvo (short for galvanometer) scanner axis is a single-axis, single-phase rotary mirror positioner mated to a reflective mirror with limited travel. A galvo scanner axis is typically paired with another similar axis within a galvo scanner.

I

Infinite Field of View (IFOV)

A feature of the Automation1 controller that commands synchronized laser spot motion. This motion is divided between galvo scanner motion at 200 kHz and linear servo motion at 20 kHz.

M

Mark and Measure

Iterative procedure where you use the galvo scanner to make fiducial marks on the workpiece. Then measure these marks with a microscope. You can use the error in these measurements to modify the galvo scanner calibration file or make a new one.

S

Servo Axis

A linear or rotary servo stage that positions either a part or the galvo scanner along a coordinate axis.

Revision History

Revision	Description
1.00	New User Guide

Index