

Getting Started with the NI SoftMotion™ Development Module for LabVIEW™

Contents

Introduction.....	2
Conventions	4
NI SoftMotion Development Module Components.....	5
System Requirements	5
Installing the NI SoftMotion Development Module.....	6
NI SoftMotion Development Module Documentation	6
Typical Motion System.....	7
Supervisory Control.....	8
Trajectory Generator.....	8
Spline Engine.....	8
Stepper Generator	8
Control Loop.....	9
I/O	9
Types of Moves	10
Arc Moves	10
Straight-Line Moves	11
Blending.....	12
Contouring	13

NI SoftMotion Development Module Examples	15
CompactRIO	15
Initialization Code	16
Trajectory Generator Loop	18
Main Loop	20
Compact FieldPoint	22
cFP Motion Loop VI	22
Application VI	24
M Series DAQ Device	26
M Series Motion Loop VI	27
Application VI	29
Where to Go for Support	32

Introduction

This manual is designed to get you started with the NI SoftMotion Development Module for LabVIEW. It includes overview information about the software as well as reference information for selected example VIs that are included with the NI SoftMotion Development Module.

The NI SoftMotion Development Module for LabVIEW provides VIs and functions that allow you to build custom motion controllers using the LabVIEW Real-Time Module and National Instruments Reconfigurable I/O (RIO) devices, data acquisition (DAQ) devices, or Compact FieldPoint (cFP).

The NI SoftMotion Development Module allows you to do path planning, trajectory generation, output control, and PID loop control. The following illustration shows several options for using the NI SoftMotion Development Module with National Instruments hardware products.

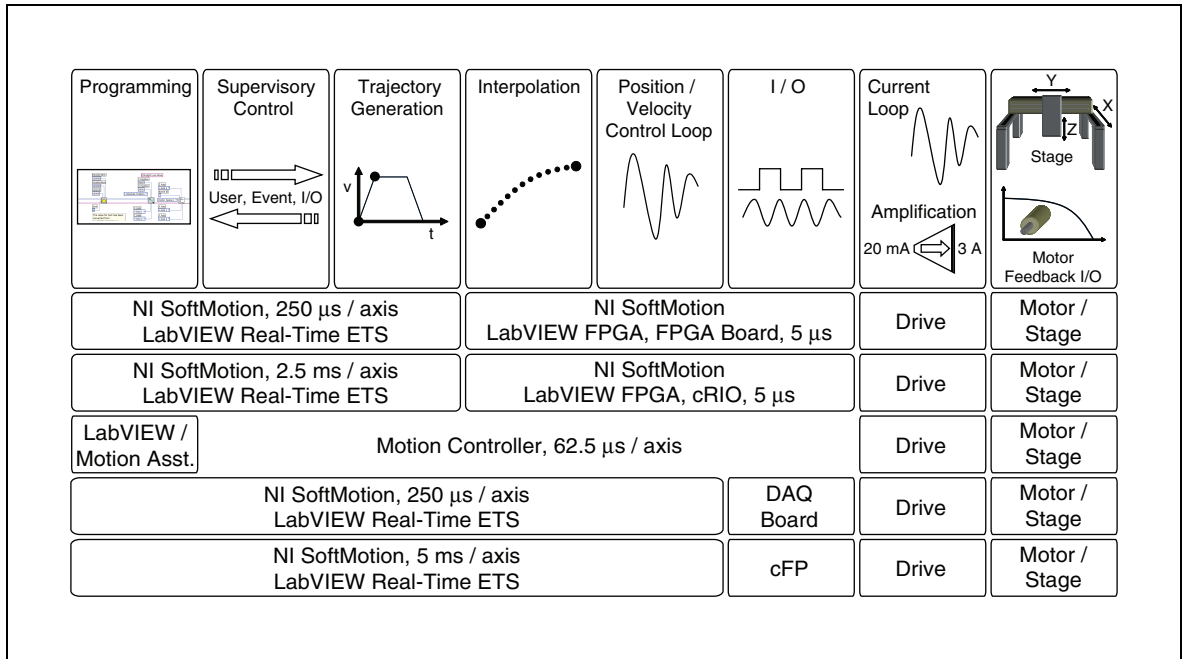


Figure 1. Possible Combinations of the NI SoftMotion Development Module and NI Hardware

Refer to the documentation for the LabVIEW Real-Time Module, the LabVIEW FPGA Module, the LabVIEW Simulation Module, and the LabVIEW Control Design Toolkit for information about system requirements, installation, and configuration of these products.

Conventions

The following conventions are used in this guide:

» The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.



This icon denotes a note, which alerts you to important information.

bold Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.

italic Italic text denotes variables, emphasis, cross-references, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.

monospace Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

NI SoftMotion Development Module Components

The NI SoftMotion Development Module includes a trajectory generator, spline engine, stepper generator, and source code for a PID control loop and encoder.

Table 1 lists each of the NI SoftMotion Development Module components, the LabVIEW platform under which each component can run, and a description of how each component is used.

Table 1. NI SoftMotion Development Module Requirements and Uses

Component	NI Platform	Use
Trajectory Generator	LabVIEW Real-Time Module	Design, compile, and run the Trajectory Generator for a Real-Time (RT) target.
Spline Engine	LabVIEW Real-Time and LabVIEW FPGA Modules	Design and compile the Spline Engine for an RT target, or download it to an FPGA target.
Stepper Generator	LabVIEW FPGA Module	Design and compile the Stepper Generator to download to an FPGA target.
PID Control Loop	LabVIEW Real-Time and LabVIEW FPGA Modules	Design and compile fixed-point PID implementation for an RT target, or download it to an FPGA target.
Encoder	LabVIEW FPGA Module	Design and compile the Encoder to download to an FPGA target.

System Requirements

Platform and system requirements for the NI SoftMotion Development Module depend on the hardware you are using. The following list contains basic system requirements that pertain to the software only. Refer to the appropriate hardware documentation for complete system requirements.

- LabVIEW Development System
- LabVIEW Real-Time Module
- LabVIEW FPGA Module



Note Refer to the *NI SoftMotion Development Module Readme* file for currently supported National Instruments software versions.

Installing the NI SoftMotion Development Module



Note To install the NI SoftMotion Development Module on a Windows Vista/XP/2000 system, you must be logged in with Administrator privileges.

1. Insert the NI SoftMotion Development Module for LabVIEW CD into the CD-ROM drive.
 - If you have autorun enabled, `autorun.exe` runs automatically.
 - If you do not have autorun enabled, double-click `autorun.exe`.
2. Follow the onscreen instructions.



Note (Windows Vista) Users may see access and security messages during installation. Accept the prompts to complete the installation.

By default, the NI SoftMotion Development Module installation program adds the following items to the listed directories in your LabVIEW installation location:

- `examples\Motion\SoftMotion` folder—Examples
- `help` folder—*NI SoftMotion Development Module for LabVIEW Help*
- `manuals` folder—*Getting Started with the NI SoftMotion Development Module for LabVIEW*



Note The NI SoftMotion Development Module installation program also installs the `NI_SoftMotion_Development_Module_ReadMe.htm` file in your `LabVIEW/readme` folder.

NI SoftMotion Development Module Documentation

In addition to this manual, the NI SoftMotion Development Module includes the *NI SoftMotion Development Module for LabVIEW Help*. This help file provides VI and function reference information for the NI SoftMotion Development Module VIs and functions. You can access this file from the **Help** menu in LabVIEW.

Typical Motion System

A motion controller is the center of a typical motion system, which consists of supervisory control, trajectory generation, spline engine, control loop, and I/O.

The controller converts high-level commands from the user into command signals used by drives to move actuators. The motion controller also monitors the system for error conditions, faults, and asynchronous events that can cause the system to change speed, direction, or start/stop the actuators. Figure 2 illustrates the parts and processes of a typical motion system.

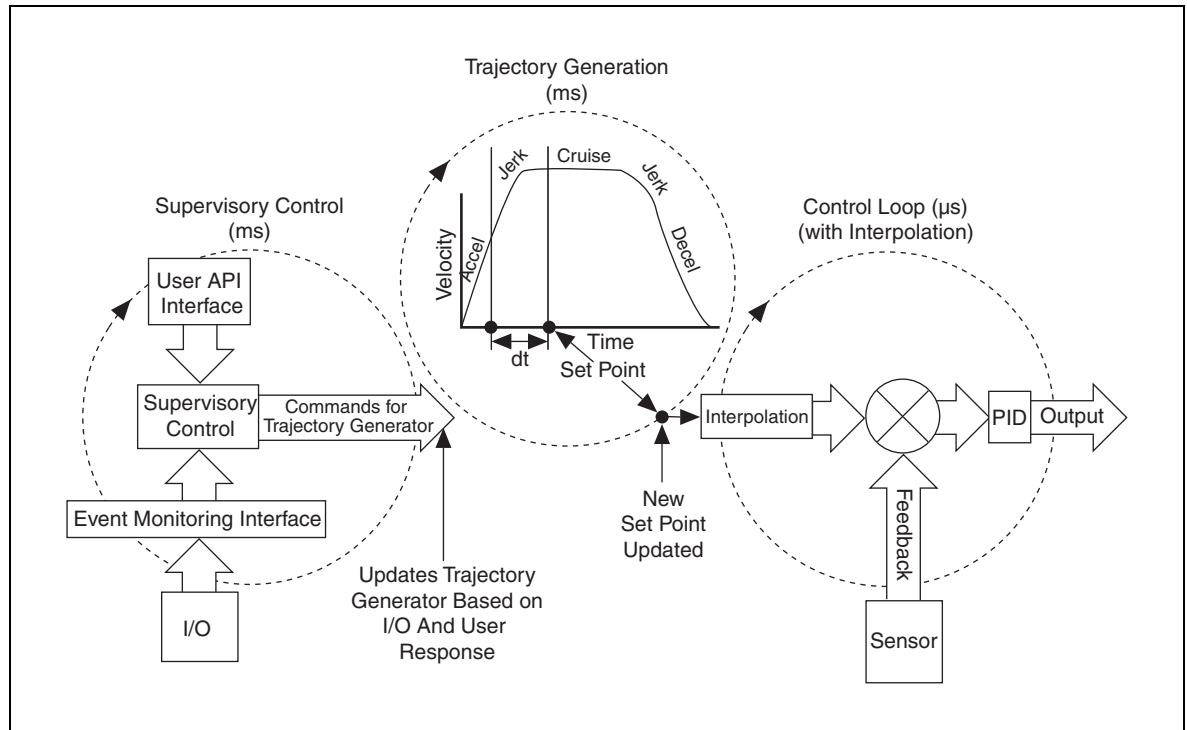


Figure 2. Three Loops of a Typical Motion Control System

Supervisory Control

The supervisory control is the main loop of the motion control system. This loop intercepts commands from the user and signals the trajectory generator to start or stop moves. The supervisory control loop also monitors all I/O needed to perform initialization tasks, such as finding the reference or origin. This loop also monitors the system for faults, and aids in synchronizing moves relative to changes in external conditions.



Note The NI SoftMotion Development Module does not contain VIs and functions for supervisory control. For examples of supervisory control, refer to the *NI SoftMotion Development Module Examples* section of this manual.

Trajectory Generator

The trajectory generator is a path planner that creates set points for the control loop. The trajectory generator creates new set points every loop period, based on move constraints provided by the user. These move constraints include the maximum velocity, maximum acceleration/deceleration, and maximum jerk that the mechanical system can tolerate. The set points created by the trajectory generator do not violate the specified move constraints.

Spline Engine

The Spline Engine function uses a cubic spline algorithm and four set points to calculate interpolated positions between two positions from the trajectory generator. You can configure the Spline Engine function to return either set point information or step data in addition to set points.

Using the Spline Engine function results in smoother motion and allows you to run the trajectory generator loop slower than the control loop.

Stepper Generator

The Stepper Generator function takes step data from the Spline Engine function and returns step and direction signals to use with stepper motors. Using the Stepper Generator function allows you to control the position of the motor and not just the velocity.

For stepper motor operation, the NI SoftMotion Development Module provides an example using the Spline Engine function with step data output and the Stepper Generator function. This example is installed to `<LabVIEW>\examples\motion\SoftMotion\StepperGeneration`.

Control Loop

The control loop creates the command signal based on the set point provided by the trajectory generator. In most cases, the control loop includes both a position and a velocity loop, but in some cases the control loop may include only a position loop.

The position is typically read from encoders, but also may be read from analog inputs. The velocity is calculated from the position values that are read. The velocity also may be read directly from a velocity sensor, such as a tachometer.

Because no feedback is required for stepper motors, the control loop converts the set point generated by the trajectory generator into stepper signals (step/direction).

For control loop implementation, the NI SoftMotion Development Module provides LabVIEW source code for an enhanced PID implementation of the control loop. The source code is installed to <LabVIEW>\examples\motion\SoftMotion\ControlLoop.

I/O

Analog and digital I/O is required to send the command signals to the drives and receive feedback from the actuators. Some servo motor drives use analog I/O to receive feedback. However, most I/O requirements for motion controllers are digital. For example, feedback from quadrature encoders requires digital I/O.

Table 2 lists the NI SoftMotion Development Module VIs you use for decoding quadrature encoder signals.

Table 2. Encoder VIs

Device	VI	Installation Location
CompactRIO	_CRIO Target Loop	<LabVIEW>\examples\Motion\SoftMotion\CompactRio\Gantry\FPGA
RIO	Control Loop (with Spline)	<LabVIEW>\examples\Motion\SoftMotion\RIO
RIO	Control Loop (with velocity feedback)	<LabVIEW>\examples\Motion\SoftMotion\RIO

Table 2. Encoder VIs (Continued)

Device	VI	Installation Location
RIO	Control Loop (with Vff and Aff)	<LabVIEW>\examples\Motion\SoftMotion\RIO
RIO	Simple Control Loop	<LabVIEW>\examples\Motion\SoftMotion\RIO

Types of Moves

The following sections describe the types of moves commonly found in motion control systems.

Arc Moves

An arc move causes a coordinate space of axes to move on a circular, spherical, or helical path. You can move a two-dimensional vector space on a circular path. You can move a three-dimensional vector space on a spherical or helical path.

Table 3 lists the NI SoftMotion Development Module function for arc moves.

Table 3. NI SoftMotion Development Module Arc Move Function

Function	Use
Arc Move Read/Write	Set or get the properties for a circular, helical, or spherical arc move. All arc moves are treated as relative to the current position. You can set or get arc move properties only from coordinates.

Refer to the *NI SoftMotion Development Module for LabVIEW Help* for more information about the Arc Move Read/Write function.

Straight-Line Moves

Straight-line moves use the desired target position to generate the move trajectory. For example, if the motor is currently at position zero, and the target position is 100, the straight-line move creates a trajectory that moves 100 counts or steps.

Table 4 lists the NI SoftMotion Development Module function for straight-line moves.

Table 4. NI SoftMotion Development Module Straight-Line Move Function

Function	Use
Straight Line Move Read/Write	Set or get parameters for a straight line move. You can read and write these properties for both axes and coordinates.

Refer to the *NI SoftMotion Development Module for LabVIEW Help* for more information about the Straight Line Move Read/Write function.

Blending

The NI SoftMotion Development Module allows you to blend moves. Blending is a software feature that automatically creates a smooth transition between consecutive moves. Moves configured for blending execute concurrently for a period of time, which results in continuous motion between moves.

Because the deceleration portion of the blended move is modified by the subsequent move, it may not reach its end point. You can change the move constraints for each blended move to create a move profile that suits your requirements. Figure 3 shows two blended moves.

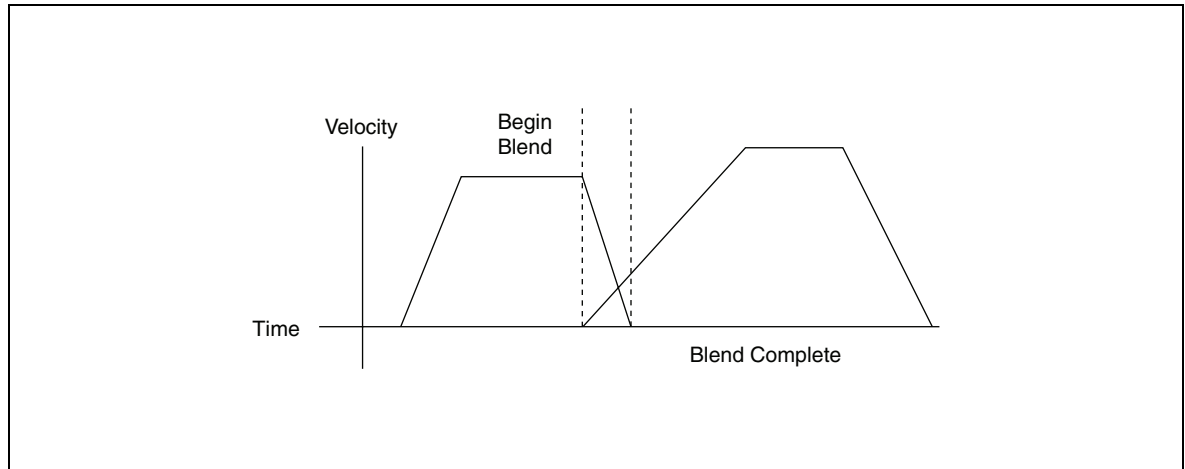


Figure 3. Blending Two Moves

Blending executes moves concurrently. When you blend the second move, both moves are executing. The first move is generating its trajectory while the second move is waiting for the blending to complete. The second move starts based on the user-specified blend mode. Blend mode options include blend at deceleration, blend after profile complete, and blend after a specified delay.

Table 5 lists the NI SoftMotion Development Module functions you use to implement blending in your application.

Table 5. NI SoftMotion Development Module Blending Functions

Function	Use
Trajectory Generator Method—Start	Execute a blend when Type is set to Blend Motion .
Trajectory Generator Method—Start Multiple	Execute a blend when Type is set to Blend Motion .
Move Constraints Read/Write	Set the Blend Delay and Blend Mode properties.
Trajectory Generator Method—Execution Data	Get the Blend Complete and Profile Complete parameters of the Status property.

Refer to the *NI SoftMotion Development Module for LabVIEW Help* for more information about blending functions.

Contouring

Contouring consists of a user-defined pattern of moves applied to an axis or a coordinate space of axes. Contoured moves are appropriate when you need a trajectory that cannot be constructed from straight lines and arcs. Instead of using the trajectory generator, the controller takes an array of position data during a contoured move and splines the data before outputting it. This calculation ensures smooth motion by creating intermediate points using a cubic spline algorithm. The basic trajectory move constraints—maximum velocity, acceleration, and deceleration—have no effect on contoured moves.

All contoured moves are relative. Motion starts from the position of the axis, or axes, at the time the contouring move starts. This is similar to the way arc moves are configured.

Table 6 lists the NI SoftMotion Development Module functions you use to implement contouring in your application.

Table 6. NI SoftMotion Development Module Contouring Functions

Function	Use
Contouring Move Read/Write	Set and get the properties on a contouring buffer.
Contouring Move Create Buffer	Create a buffer for contouring, and return a handle to the buffer.
Contouring Move Delete Buffer	Delete the contouring buffer.
Contouring Move Method—Check Buffer Space	Retrieve the number of elements available for updating.
Contouring Move Method—Enable	Enable or disable a buffer.
Contouring Move Method—Stop	Signal the final iteration of a continuous buffer, and stop the move once the end of the buffer is reached.
Contouring Move Method—Update Points	Replace points previously written to a buffer with newer points.
Contouring Move Method—Write Points	Write points to the buffer for the first time.

Refer to the *NI SoftMotion Development Module for LabVIEW Help* for more information about contouring functions.

NI SoftMotion Development Module Examples

This section outlines selected examples installed with the NI SoftMotion Development Module for LabVIEW. You can browse or search for additional examples using the NI Example Finder. Refer to the *Finding Example VIs* topic of the *LabVIEW Help* for more information about using the NI Example Finder.

Refer to the *NI SoftMotion Development Module Reference Architecture* in the NI Developer Zone at ni.com/zone for a VI template that simplifies programming with the NI SoftMotion Development Module. Go to ni.com/info and enter the Info Code `rdsmat` to access the example as well as instructions on how to implement it in your applications.

The CompactRIO example contains a VI that runs on the real-time system and a VI that runs on the FPGA target. In this example, the real-time VI performs the same function as the Motion Loop VI in the previous examples. The FPGA VI contains the stepper generator as well as improved timing code to eliminate clock drift between the real-time system and the FPGA target.

The examples for Compact FieldPoint and M Series DAQ consist of a Motion Loop VI and a set of application VIs. In these examples, the Motion Loop VI performs the function of a motion controller in a typical motion control system. This VI implements the code required to interact with I/O, read in trajectory generator information, calculate the move profile, write the move profile information, and calculate the control output. The application VIs interact with the Motion Loop VI to change the state of the Motion Loop VI and get the status from the Motion Loop VI.

CompactRIO

The following sections describe the CompactRIO stepper generation example installed with the NI SoftMotion Development Module. This example shows how to create a single axis move for stepper motors using the LabVIEW FPGA module with an improved synchronization technique between the RT host and the FPGA target. This technique uses interrupts to synchronize the FPGA and RT loops, eliminating clock drift.



Note In this example, the spline engine and stepper generator run on the FPGA, and the trajectory generator runs on the RT system. Only the trajectory generation and API sections of the example are shown.

Initialization Code

Figure 4 shows the block diagram of the initialization section. This section of the VI initializes a front panel button, configures and initializes the FPGA target, and configures and opens a reference to the trajectory generator.

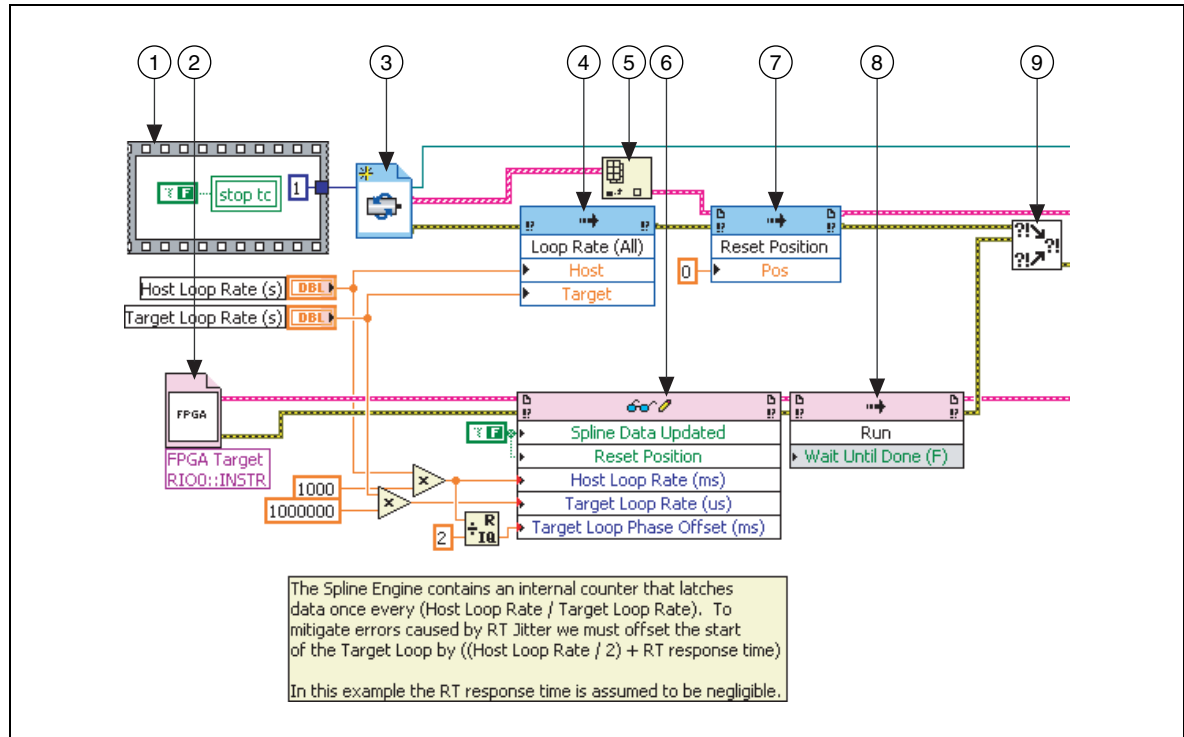


Figure 4. CompactRIO Stepper Generation VI Initialization Code

Table 7. Description of CompactRIO Stepper Generation VI Initialization Code

Callout Number	VI, Function, or Variable Name and Description
1	Sequence structure and global variable used to set the stop button to FALSE.
2	LabVIEW Open FPGA VI Reference VI determines what I/O you use on the CompactRIO device, which includes analog and digital I/O. This VI starts the fpga_stepper VI, which implements the spline engine, stepper generator, and timing. Opening this reference allows the real-time VI to access all the registers, or controls and indicators, on the fpga_stepper VI.
3	Trajectory Generator Create function opens a reference to the trajectory generator. Always execute this function prior to making any calls into the trajectory generator.
4	Trajectory Generator Method—Loop Rate (All) sets up the loop rate for the trajectory generator and motion loop.
5	LabVIEW Index Array function used to send one axis to the trajectory generator.
6	LabVIEW Read/Write Control sends the loop rates for the FPGA program and initializes the controls.
7	Trajectory Generator Method—Reset Position resets the trajectory generator position to 0.
8	LabVIEW FPGA Invoke Method starts the FPGA VI on the FPGA target.
9	LabVIEW Merge Errors VI. Use this VI to merge error I/O clusters from different functions. This VI first looks for errors among error in 1 , error in 2 , and error in 3 , then error array in , and reports the first error found. If the VI finds no errors, it looks for warnings, and returns the first warning found. If the VI finds no warnings, it returns <code>no error</code> .

Trajectory Generator Loop

Figure 5 shows the trajectory generator loop of the CompactRIO stepper generation VI. This loop repeatedly cycles through a set of given positions to produce individual setpoints to send to the stepper motor.

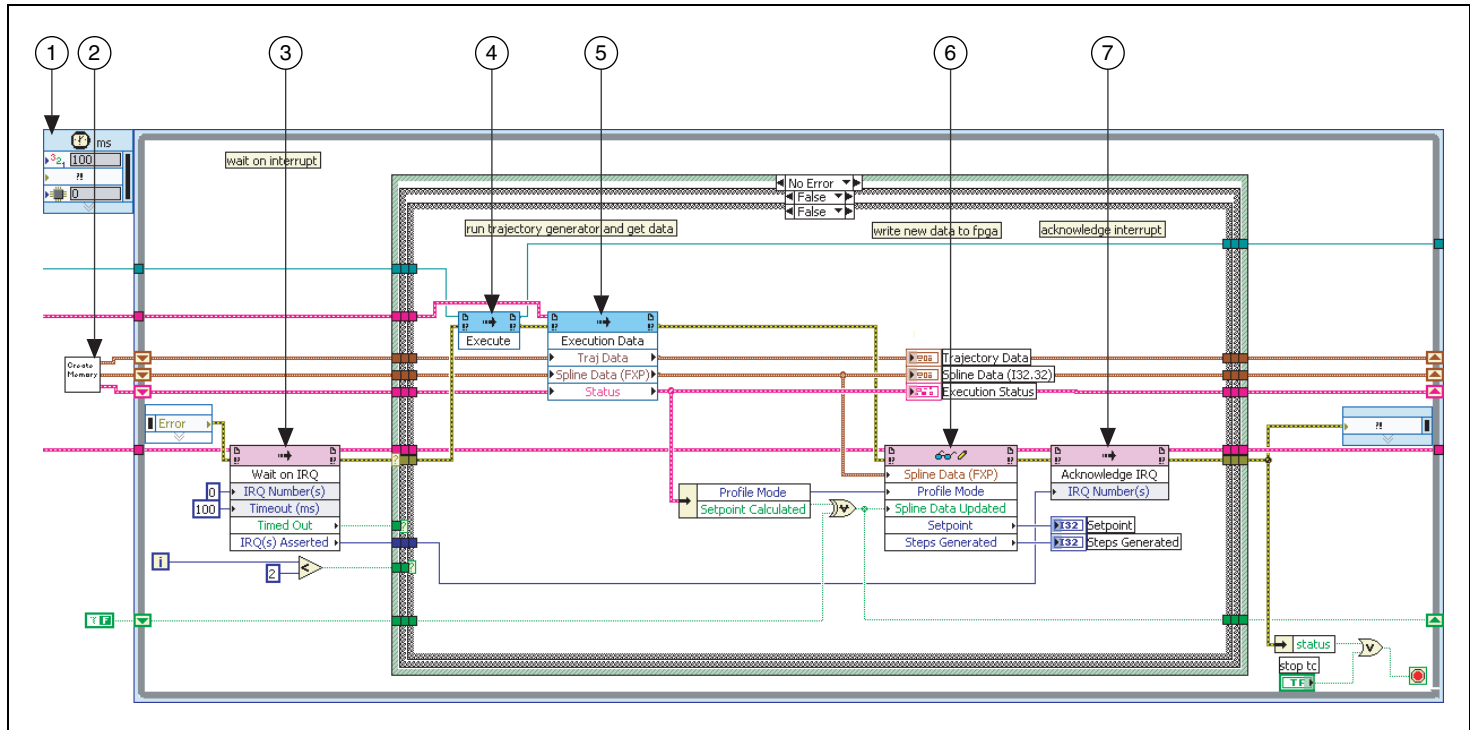


Figure 5. CompactRIO Stepper Generation VI Trajectory Generator Loop

Table 8. Description of CompactRIO Stepper Generation VI Trajectory Generator Loop

Callout Number	VI, Function, or Variable Name and Description
1	LabVIEW Timed Loop. Use this node to set up loop timing.
2	User-defined VI allocates memory on the CompactRIO device. This VI improves the performance of the application because it allocates memory only once rather than each time the VI loops. Use this VI to allocate memory for the following items: <ul style="list-style-type: none">• Execution status data• Trajectory data• Spline data
3	LabVIEW FPGA Invoke Method waits for an interrupt from the FPGA.
4	Trajectory Generator Method—Execute method runs calculations for the axis.
5	Trajectory Generator Method—Execution Data method updates the trajectory generator information for the axis.
6	LabVIEW Read/Write Control. Use this node to send the latest trajectory data to the FPGA.
7	LabVIEW FPGA Invoke Method acknowledges the interrupt from the FPGA.

Main Loop

Figure 6, Figure 7, and Figure 8 show the main loop of the CompactRIO stepper generation VI. This loop contains the code used to move the stepper motor based on user input. Figure 6 shows the default case, which updates the front panel with current velocity and position information.

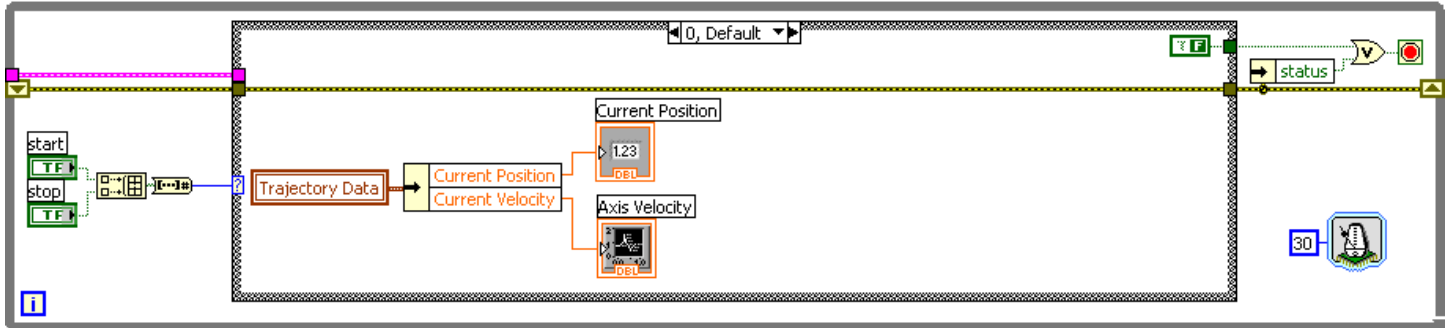


Figure 6. CompactRIO Stepper Generation VI Main Loop Default Case

Figure 7 shows the code that executes when the user clicks the Start button on the front panel.

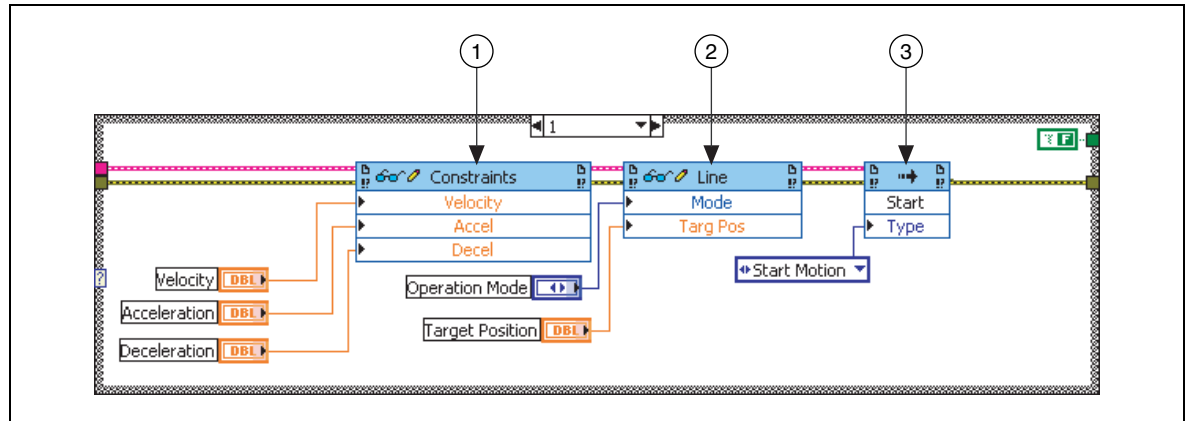


Figure 7. CompactRIO Stepper Generation VI Main Loop Start Case

Table 9. Description of CompactRIO Stepper Generation VI Main Loop Start Case

Callout Number	VI, Function, or Variable Name and Description
1	Move Constraints Read/Write updates the trajectory generator with new velocity, acceleration, and deceleration values.
2	Straight Line Move Read/Write sets the target position and operation mode for the trajectory generator.
3	Trajectory Generator Method—Start method starts the move.

Figure 8 shows the code that executes when the user clicks the Stop button on the front panel.

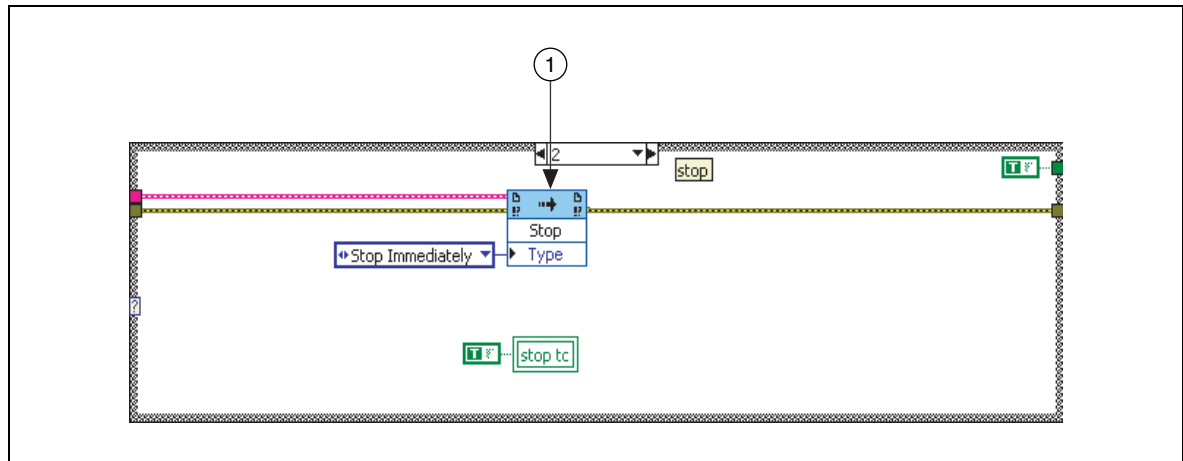


Figure 8. CompactRIO Stepper Generation VI Main Loop Stop Case

Table 10. Description of CompactRIO Stepper Generation VI Main Loop Start Case

Callout Number	VI, Function, or Variable Name and Description
1	Trajectory Generator Method—Stop method stops the move.

Compact FieldPoint

The following sections describe the Compact FieldPoint example installed with the NI SoftMotion Development Module. This example uses a Compact FieldPoint device to independently control two motors.

cFP Motion Loop VI

Figure 9 shows the block diagram of the cFP Motion Loop VI, which implements a motion controller for two axes of motion on a Compact FieldPoint device.

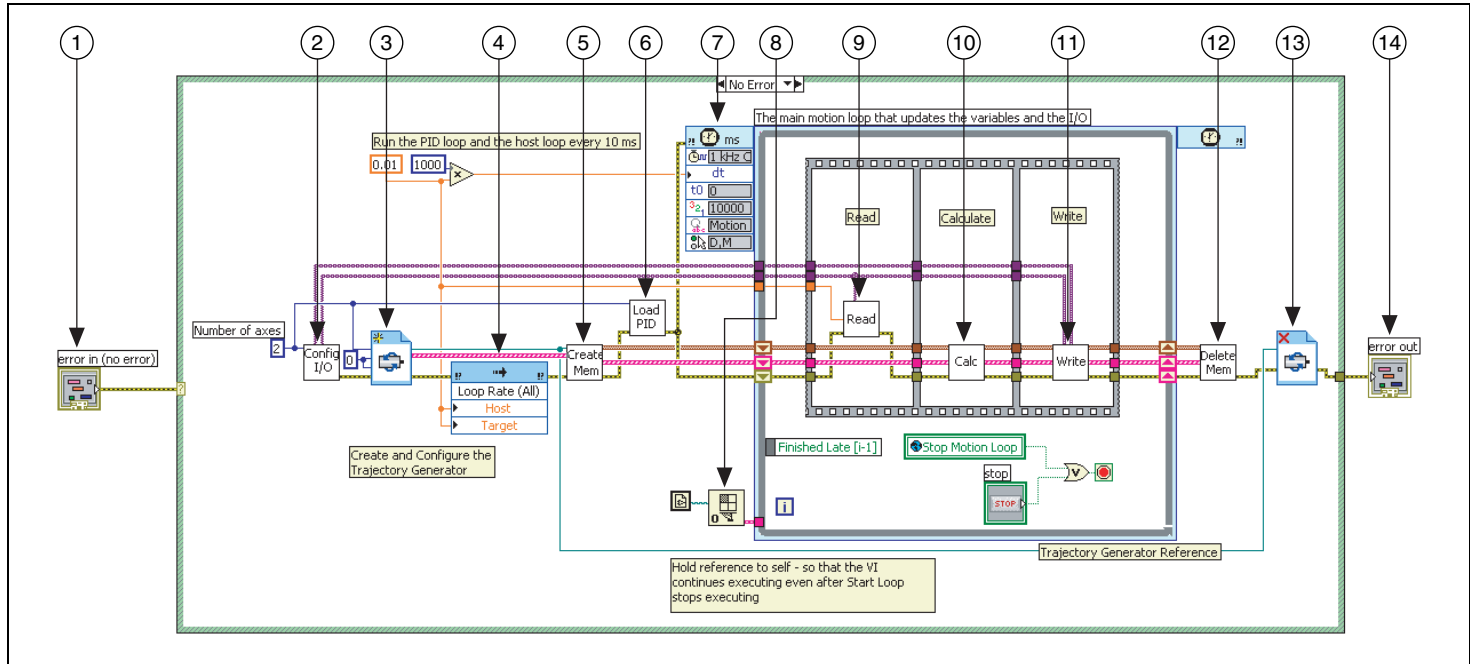


Figure 9. cFP Motion Loop VI Block Diagram

Table 11. Description of the cFP Motion Loop VI Block Diagram

Callout Number	VI, Function, or Variable Name and Description
1	LabVIEW error in control.
2	User-designed VI determines what I/O to use on the Compact FieldPoint device, such as encoders, analog devices, and so on.
3	Trajectory Generator Create function opens a reference to the trajectory generator. Always execute this function prior to making any calls into the trajectory generator.
4	Trajectory Generator Method—Loop Rate method sets up the loop rate for the trajectory generator and motion loop.
5	<p>User-designed VI allocates memory on the Compact FieldPoint device. This VI improves the performance of the application because it allocates memory only once rather than each time the VI loops. Use this VI to allocate memory for the following items:</p> <ul style="list-style-type: none">• Execution status data• Axis data• Trajectory data• Array of axis handles• Additional arrays that contain other state data• Control output values• PID parameters
6	User-designed VI configures the PID parameters for the axes.
7	LabVIEW Timed Loop. Use this node to set up loop timing.
8	LabVIEW Open VI Reference VI opens a reference to the VI specified by the path wired in.
9	User-designed VI reads in data from encoders and global variables.

Table 11. Description of the cFP Motion Loop VI Block Diagram (Continued)

Callout Number	VI, Function, or Variable Name and Description
10	User-designed VI executes the NI SoftMotion Development Module trajectory generator and calls the control loop to calculate command output values.
11	User-designed VI writes the command output values to the I/O on the Compact FieldPoint device.
12	User-designed VI frees the memory you allocated on the Compact FieldPoint device.
13	The Trajectory Generator Delete function frees the memory allocated for the trajectory generator you created.
14	LabVIEW error out control.

Application VI

Figure 10 shows the block diagram for the cFP Main VI. This VI creates two independent straight-line moves using two different axes.



Note For this example to work properly, the cFP Motion Loop VI must be running in the background. You can synchronize the Motion Loop VI with the Application VI in one of several ways. For example, you can use a VI from the **Advanced»Synchronization»Rendezvous** palette. Or, you can use the VI Server to dynamically run the VI. For information about synchronizing VIs, refer to the *Rendezvous VIs* topic or the *VI Server* topic in *LabVIEW Help*. Also, refer to the *What Is a Rendezvous?* article in the NI KnowledgeBase at ni.com/KB and/or *Dynamically Load and Run a VI without Opening its Front Panel Using VI Server and Rendezvous VIs* article on the NI Developer Zone at ni.com/zone.

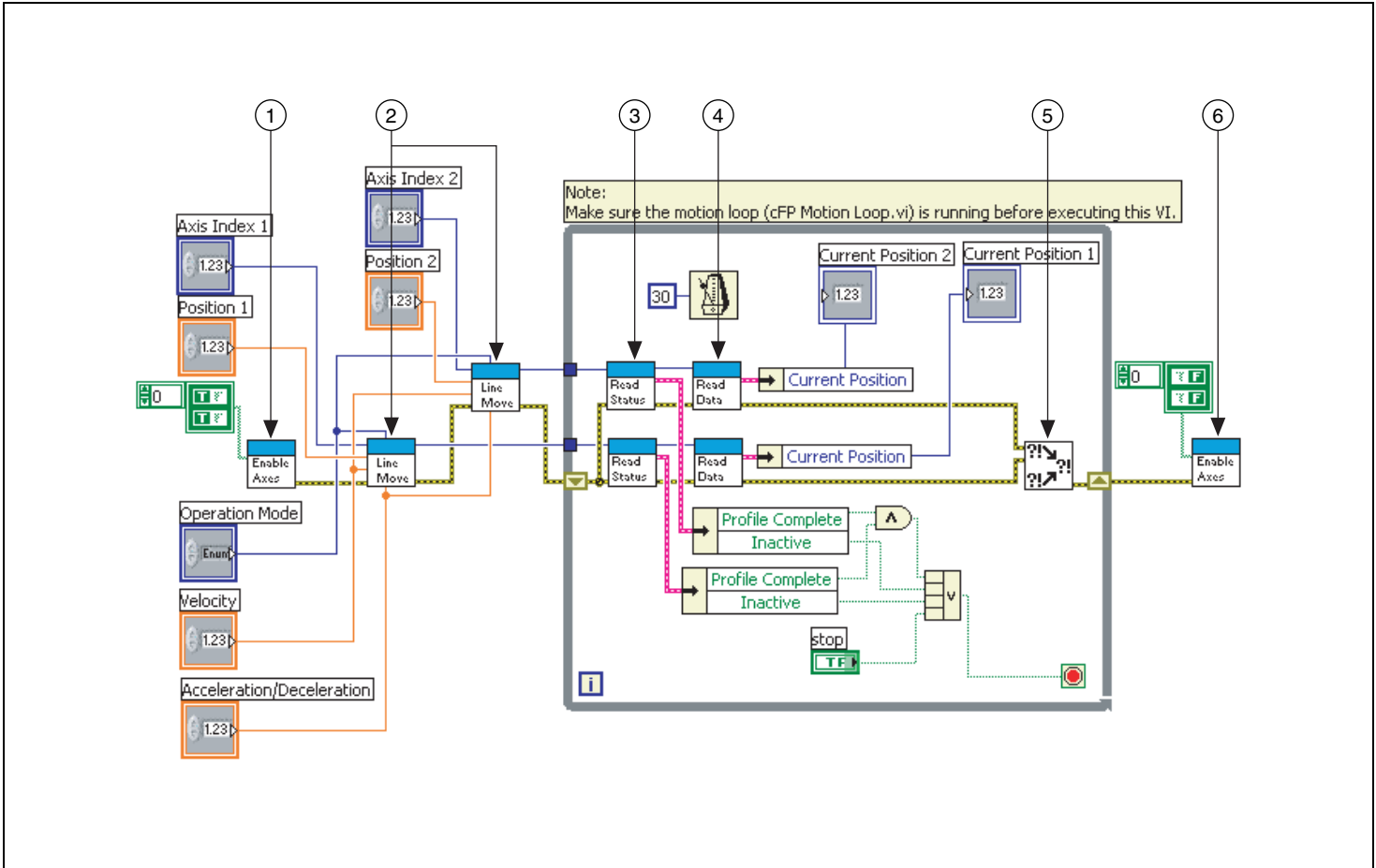


Figure 10. cFP Main VI Block Diagram

Table 12. Description of the cFP Main VI Block Diagram

Callout Number	VI, Function, or Variable Name and Description
1	User-defined VI enables the axes.
2	User-defined VI specifies the position, acceleration, deceleration, and operation mode for a particular axis.
3	User-defined VI reads the execution status for the specified axis.
4	User-defined VI reads the axis data for the specified axis.
5	LabVIEW Merge Errors VI. Use this VI to merge error I/O clusters from different functions. This VI first looks for errors among error in 1 , error in 2 , and error in 3 , then error array in , and reports the first error found. If the VI finds no errors, it looks for warnings, and returns the first warning found. If the VI finds no warnings, it returns <code>no error</code> .
6	User-defined VI disables the axes.

M Series DAQ Device

The following sections describe the M Series DAQ Device Gantry example installed with the NI SoftMotion Development Module. This example implements a motion controller for three axes using two M Series DAQ devices. Two of the axes in the example are controlled as an xy coordinate. The third axis is controlled individually as the z-axis. These three axes together constitute a gantry system.

M Series Motion Loop VI

Figure 11 shows the block diagram of the M Series Motion Loop VI. This VI implements a motion controller for one coordinate, consisting of two axes, and one independent axis on the M Series DAQ devices.

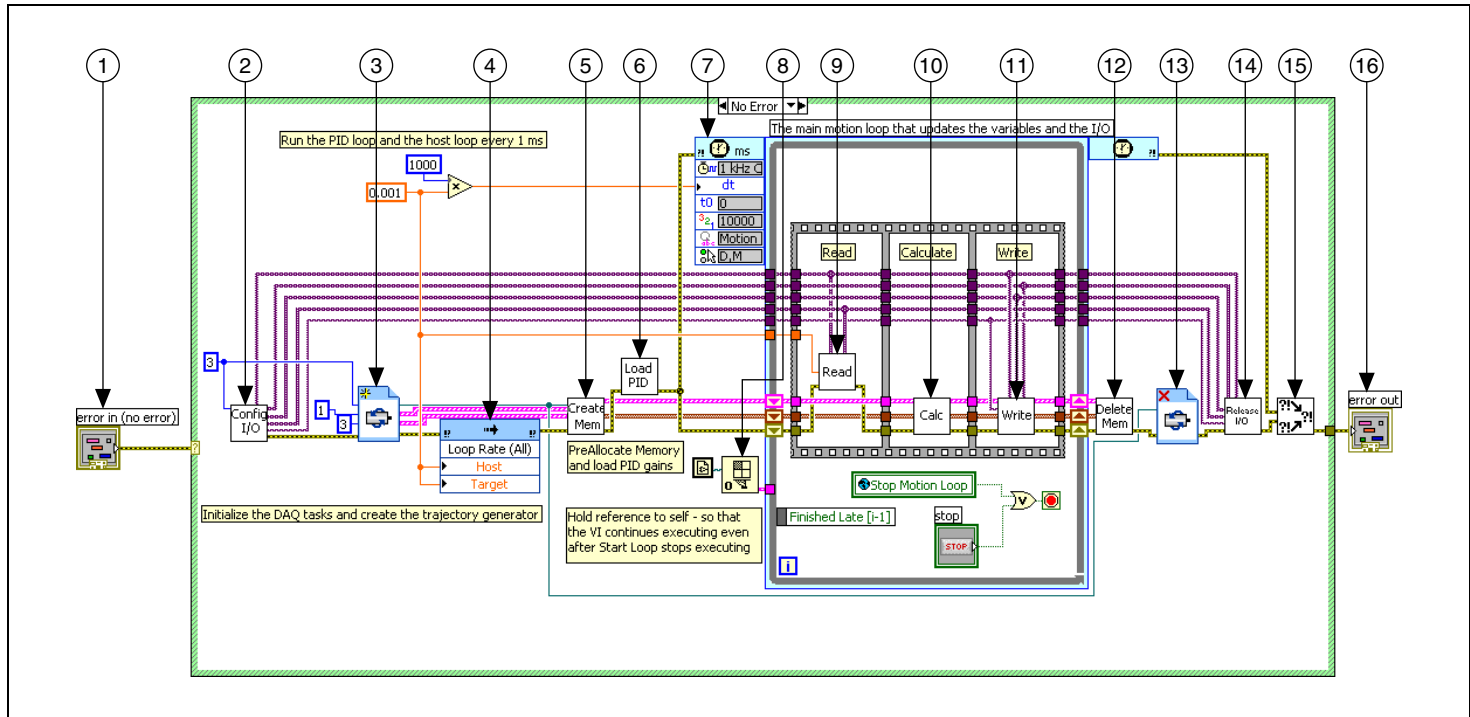


Figure 11. M Series Motion Loop VI Block Diagram

Table 13. Description of the M Series Device Motion Loop VI Block Diagram

Callout Number	VI, Function, or Variable Name and Description
1	LabVIEW error in control.
2	User-defined VI determines what I/O to use on the M Series DAQ devices, such as encoders and analog devices.
3	Trajectory Generator Create function opens a reference to the trajectory generator. Always execute this function prior to making any calls into the trajectory generator.
4	Trajectory Generator Method—Loop Rate Method sets up the loop rate for the trajectory generator and motion loop.
5	<p>User-defined VI allocates memory on the DAQ M Series devices. This VI improves the performance of the application because it allocates the memory only once rather than each time the VI loops. Use this VI to allocate memory for the following items:</p> <ul style="list-style-type: none">• Execution status data• Axis data• Trajectory data• Array of axis handles• Additional arrays that contain other state data• Control output values• PID parameters
6	User-defined VI configures the PID parameters for the axes.
7	LabVIEW Timed Loop node. Use this node to set up loop timing.
8	LabVIEW Open VI Reference VI opens a reference to the VI specified by the path wired in.
9	User-defined VI reads in data from encoders and global variables.

Table 13. Description of the M Series Device Motion Loop VI Block Diagram (Continued)

Callout Number	VI, Function, or Variable Name and Description
10	User-defined VI executes the NI SoftMotion Development Module trajectory generator and calls the control loop to calculate command output values.
11	User-defined VI writes the command output values to the I/O on the M Series DAQ devices.
12	User-defined VI frees the memory you allocated.
13	Trajectory Generator Delete function frees the memory allocated for the trajectory generator you created.
14	User-defined VI releases the I/O you previously configured.
15	LabVIEW Merge Errors VI. Use this VI to merge error I/O clusters from different functions. This VI first looks for errors among error in 1 , error in 2 , and error in 3 , then error array in , and reports the first error found. If the VI finds no errors, it looks for warnings, and returns the first warning found. If the VI finds no warnings, it returns <code>no error</code> .
16	LabVIEW error out control.

Application VI

Figure 12 shows the block diagram of the M Series Gantry Cycle VI. This VI repeatedly cycles through a set of given positions resulting in coordinate motion along the x- and y-axes and vertical motion along the z-axis.



Note For this example to work properly, the M Series Motion Loop VI must be running in the background. You can synchronize the Motion Loop VI with the Application VI in one of several ways. For example, you can use a VI from the **Advanced»Synchronization»Rendezvous** palette. Or, you can use the VI Server to dynamically run the VI. For information about synchronizing VIs, refer to the *Rendezvous VIs* topic or the *VI Server* topic in *LabVIEW Help*. Also, refer to the *What Is a Rendezvous?* article in the NI KnowledgeBase at ni.com/kb and/or *Dynamically Load and Run a VI without Opening its Front Panel Using VI Server and Rendezvous VIs* article on the NI Developer Zone at ni.com/zone.

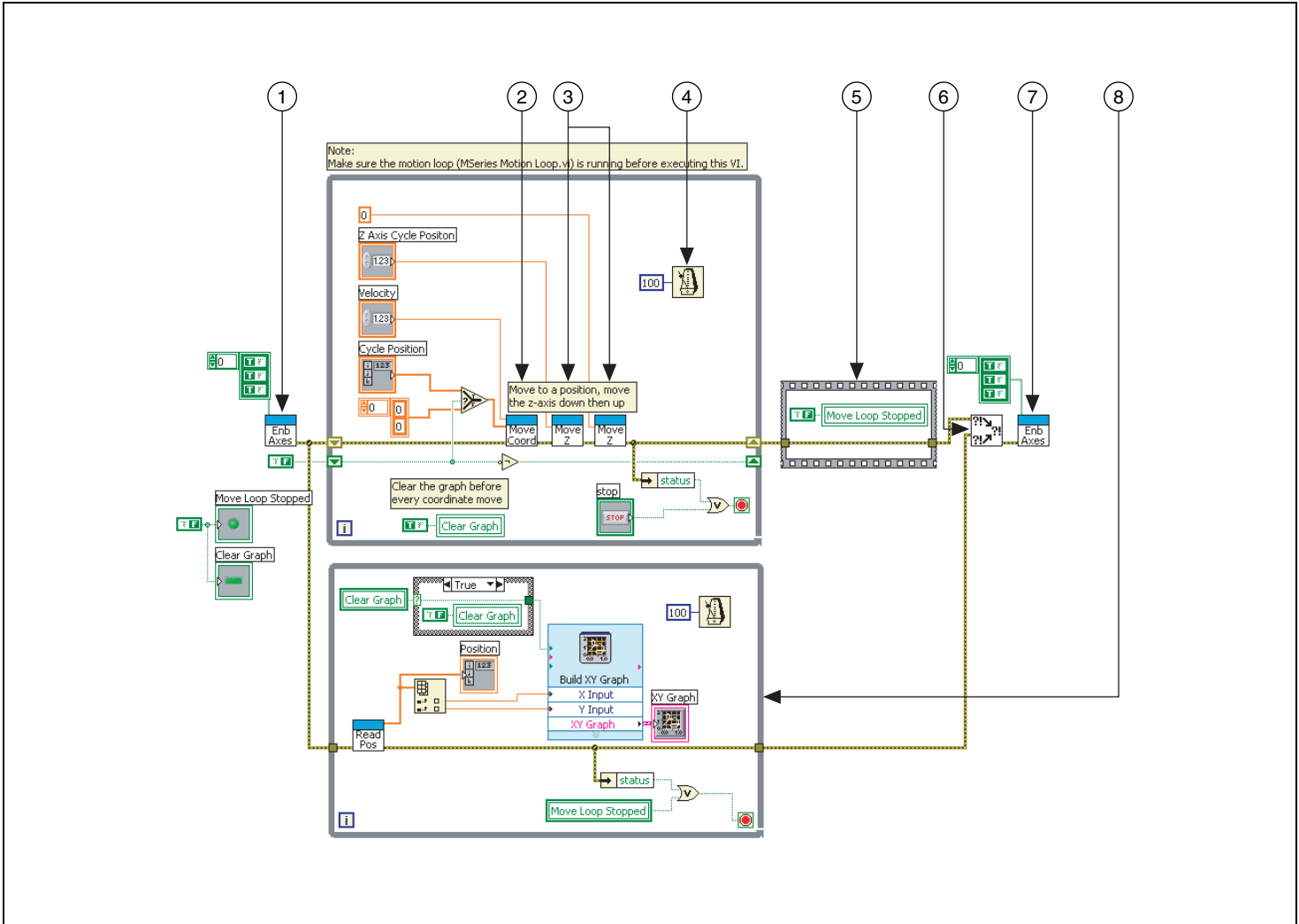


Figure 12. M Series Gantry Cycle VI Block Diagram

Table 14. Description of M Series Gantry Cycle VI Block Diagram

Callout Number	VI, Function, or Variable Name and Description
1	User-defined VI enables the axes.
2	User-defined VI specifies the following move constraints: <ul style="list-style-type: none">• Target position• Velocity• Acceleration rate• Deceleration rate• Jerk values for both the acceleration and deceleration
3	User-defined VI moves the motor up and back on the z-axis after reaching the position specified in the previous VI.
4	LabVIEW Wait Until Next ms Multiple VI. Use this VI to specify a delay for the While Loop.
5	Case statement and global variable used to determine if motion has stopped.
6	LabVIEW Merge Errors VI. Use this VI to merge error I/O clusters from different functions. This VI first looks for errors among error in 1 , error in 2 , and error in 3 , then error array in , and reports the first error found. If the VI finds no errors, it looks for warnings, and returns the first warning found. If the VI finds no warnings, it returns <code>no error</code> .
7	User-defined VI disables the axes.
8	While Loop to read the current axes position using a user-defined VI, and then graph the position using the Build XY Graph Express VI.

Where to Go for Support

The National Instruments Web site is your complete resource for technical support. At ni.com/support you have access to everything from troubleshooting and application development self-help resources to email and phone assistance from NI Application Engineers.

National Instruments corporate headquarters is located at 11500 North Mopac Expressway, Austin, Texas, 78759-3504. National Instruments also has offices located around the world to help address your support needs. For telephone support in the United States, create your service request at ni.com/support and follow the calling instructions or dial 512 795 8248. For telephone support outside the United States, contact your local branch office:

Australia 1800 300 800, Austria 43 662 457990-0, Belgium 32 (0) 2 757 0020, Brazil 55 11 3262 3599, Canada 800 433 3488, China 86 21 5050 9800, Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 358 (0) 9 725 72511, France 01 57 66 24 24, Germany 49 89 7413130, India 91 80 41190000, Israel 972 3 6393737, Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 3451 3400, Lebanon 961 (0) 1 33 28 28, Malaysia 1800 887710, Mexico 01 800 010 0793, Netherlands 31 (0) 348 433 466, New Zealand 0800 553 322, Norway 47 (0) 66 90 76 60, Poland 48 22 3390150, Portugal 351 210 311 210, Russia 7 495 783 6851, Singapore 1800 226 5886, Slovenia 386 3 425 42 00, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 (0) 8 587 895 00, Switzerland 41 56 2005151, Taiwan 886 02 2377 2222, Thailand 662 278 6777, Turkey 90 212 279 3031, United Kingdom 44 (0) 1635 523545