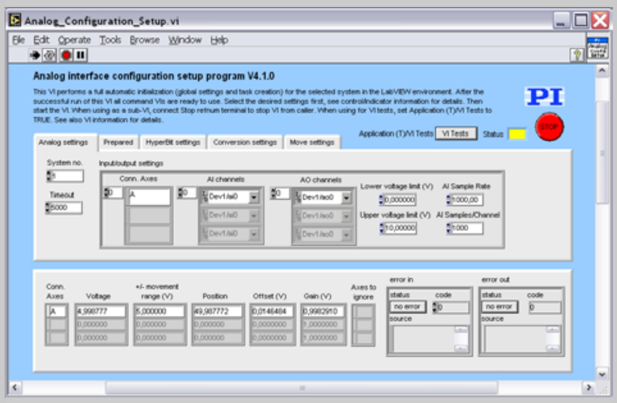
Piezo · Nano · Positioning

Analog Controller **LabView Driver Library**

Release: 5.2.2 Date: 2007-04-05


Analog_Configuration_Setup.vi

Analog interface configuration setup program V4.1.0

This VI performs a full automatic initialization (global settings and task creation) for the selected system in the LabVIEW environment. After the successful run of this VI all command VIs are ready to use. Select the desired settings first, see control/indicator information for details. Then start the VI. When using as a sub-VI, connect Stop retnum terminal to stop VI from caller. When using for VI tests, set Application (T)/VI Tests to TRUE. See also VI information for details.

This document describes software for use with the following product(s):

- **PI Analog Controllers**
  PI controllers having Analog Inputs

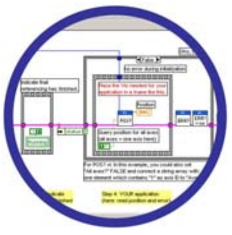
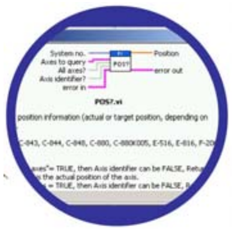



Table of Contents

Release: 5.2.2

File:Analog_GCSLabVIEW_ PZ181E _522.doc, 1873920 Bytes

0. Disclaimer

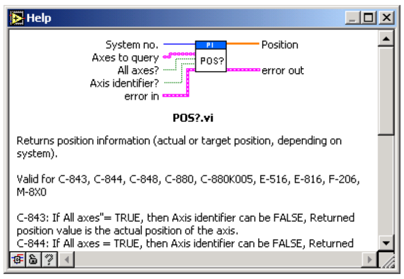
This software is provided "as is". Physik Instrumente (PI) does not guarantee that this software is free of errors and will not be responsible for any damage arising from the use of this software. The user agrees to use this software on his own responsibility.

1. Introduction

The LabVIEW software consists of a collection of virtual instrument (VI) drivers. All functionality involves invoking one or more VIs with the appropriate parameter and global variable settings.

These VIs are provided to ease the task of programming your application. They, and the accompanying documentation, assume a prior knowledge of proper LabView programming techniques. The provided "Simple Test" and "Configuration Setup" VIs help to solve the essential initialization steps, but are not intended to provide an out-of-the-box, universal solution to a particular application.

To minimize the need for consulting the manual during programming, each VI comes with a detailed VI description that appears in the *Context Help* window when you move the cursor over the VI icon. Use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window.

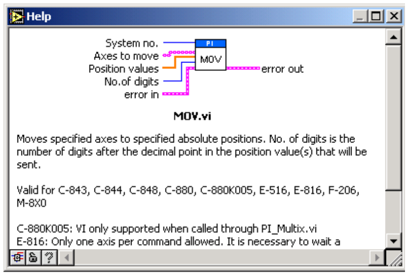


POS?.vi

Returns position information (actual or target position, depending on system).

Valid for C-843, C-844, C-848, C-880, C-880K005, E-516, E-816, F-206, M-8X0

C-843: If All axes"= TRUE, then Axis identifier can be FALSE, Returned position value is the actual position of the axis.
C-844: If All axes = TRUE, then Axis identifier can be FALSE, Returned



MOV.vi

Moves specified axes to specified absolute positions. No. of digits is the number of digits after the decimal point in the position value(s) that will be sent.

Valid for C-843, C-844, C-848, C-880, C-880K005, E-516, E-816, F-206, M-8X0

C-880K005: VI only supported when called through PI_Multix.vi
E-816: Only one axis per command allowed. It is necessary to wait a

LabVIEW 7.1 or higher and NI-VISA 3.6 or higher must be installed prior to using this driver set. To control an analog system, DAQmx 8.3 or higher must also be installed.

1.1. PI General Command Set (GCS)

This VI driver set supports the *PI General Command Set*, which is based on ASCII communication with well-defined commands and replies. This makes it possible to control different PI systems, such as the *E-516 Display Module* or the *C-880 Multi-Axis Controller*, with only one driver set simply by "wiring" the correct command parameters to the associated VIs.

Translation Libraries

To control PI systems with a native command set that is not compatible with the *PI General Command Set*, e.g. the *E-710 Digital Piezo Controller* or the *C-843 Motion Control Board*, controller-specific libraries are used. Each such library translates *PI General Command Set* commands to the controller's native language. There is also a universal library which adds this functionality: GCSTranslator.dll; it must be installed on the computer in the GCS_LabVIEW\Low Level folder, no matter whether the system being controlled is *PI General Command Set* compatible or not.

For these and certain other systems (such as PC add-on cards), the required system-specific DLLs and data files (e.g. PIStages.dat) must be properly installed. If you install this driver set from within the setup program of the PI software CD ROM, this installation is done automatically. If you want to install this driver set manually, please run “GCSLibrarySetup.exe” from the CD-ROM that came with your system. This setup tool makes sure that all necessary libraries and their data files are correctly registered in the Windows™ environment and can be found by the GCS drivers (if LabVIEW still cannot find PIStages.dat, it may be because it is marked read-only. To see, open Microsoft Explorer, right-click the file PIStages.dat and select *Properties*. Make sure that the *read-only* attribute is not checked.)

To control a system using an analog interface DAQmx 8.3 and a DAQmx-compatible National Instruments DAC card must be installed.

Once the libraries and data files for the system to control are installed, this LabVIEW driver set can be used to control a non-GCS-compatible system just like any GCS-compatible system, and PCI/ISA-based controller boards by selecting “DLL” as communication interface (see Section "First Steps for GCS-Compatible PI Controllers” on p. 7 and the “XXXX_Configuration_Setup.vi” (with XXXX being the PI product number of your system) in section 3).

Units and GCS

The GCS system uses physical units of measure. Most controllers and GCS software have default conversion factors chosen to convert hardware-dependent units (e.g. encoder counts) into mm or degrees, as appropriate. These defaults are generally taken from a database of stages that can be connected. The direction of motion associated with positive and negative relative moves can also be controlled by parameter settings. In some cases an additional scale factor can be applied, making a second physical unit available without overwriting the conversion factor for the first. It is also sometimes possible to enter a conversion factor as numerator and denominator of a fraction, reducing the number of digits and outside calculations needed for high-precion entry of gearhead system values. See the DFF.vi and SPA.vi command descriptions (if supported by your PI controller), taking special note of the sections referring specifically to your controller.

1.2. Scope of This Manual

This manual covers only VIs which can be used with the product with which it came.

1.3. VI Structure

The folder structure of the LabVIEW drivers consists of the main folder “GCS_LabVIEW” with the sub-folder “Low Level”.

The main folder “GCS_LabVIEW" contains a terminal VI (for command based systems), a configuration VI (XXXX_Configuration_Setup.vi with XXXX being the PI product number of your system), a simple test VI, and, if available, several sample programs.

The sub-folder “Low Level” contains VIs for the following functions:

- Establishing communication with different PI systems which support the PI General Command Set via RS-232, GPIB or TCP/IP interfaces, or with analog systems
- Defining the parameter IDs of the connected axes

- Sending and receiving ASCII characters to/from the specified system or setting and reading voltages for an analog system
- Sending system-specific commands (system-specific commands are separated into function-specific LLBs).

Additionally, the sub-folder “Low Level” contains GCSTranslator.dll.

Following the data flow concept of LabVIEW, all VIs have their wiring inputs on the left side and their wiring outputs on the right side of each connector pane. For quick integration, this **connector pane** in most cases has the following pattern:

1					15
2	7	9	11	13	16
3					17
4					18
5	8	10	12	14	19
6					20

The terminals are assigned as follows (if the mentioned, control/indicator is present in one of the supplied libraries):

1	System number
2	Optical board, Interface, or other main input control
3	Axes to query, Affected axes, Number of systems, or other main input control
4	All axes?, Invert order?, or other main input control
5	Axis identifier?, No. of digits, or other main input control
6	Error in
7	Parameter number, Without axis ID?, or other input control
8	Step size, or other input control
9	AA step size, or other input control
10	Input control
11	Input control or output indicator
12	Input control or output indicator
13	Input control or output indicator
14	Input control or output indicator
15	Hidden error, Connected axes, String read, or other main output indicator
16	Axes to query out, Bytes read, or other main output indicator
17	No. of rows, or other main output indicator
18	Output indicator
19	Output indicator
20	Error out

Also note that this driver set does not use the standard LabVIEW error numbers recommended by National Instruments, but rather those used by PI controllers. As a result, the error texts displayed by LabVIEW will not describe the error accurately. Use “GCSTranslateError.vi” to get the description of a PI GCS error number. Some VIs use an additional indicator Controller error to indicate that the selected system has been queried for a controller error with „ERR?“ and reported an error number ≠ zero.

See also chapter 5 on p. 72 for a summary of error numbers produced by this driver set.

In LabVIEW, uncheck *Enable automatic error handling dialogs* in *Tools→Options→New and Changed in 7.x* to prevent that LabVIEW suspends execution and displays an error dialog box for any error that occurs during the execution of the VIs.

Important:

Before running any VIs to control a connected system, “**XXXX_Configuration_Setup.vi**” (located in the main folder, with XXXX being the PI product number of your system) must be run. This initialization VI performs all necessary steps automatically:

1. It opens the communications port,
2. It defines the IDs for the connected axes,
3. It references the connected stages (if appropriate), depending on if the controller requires a referencing before axes can be moved and on your custom settings,
4. It defines the controller name.

After these steps all parameters are saved into global variables, so that other VIs invoked during the same LabView session can access this data at runtime.

As the initialization is a complex procedure which uses a large number of sub-VIs, **XXXX_Configuration_Setup.vi** is password-protected, meaning that you cannot see or modify the diagram. In this way, the full initialization is packed into one single and fully tested procedure which you simply insert into your own application program. For security reasons as well as your convenience, we recommend that you not modify this VI.

For testing a PI system using a command-based interface, the easiest method is to call “PI Terminal.vi”, which is located in the ”GCS_LabVIEW” main folder. This is a ”stand-alone” routine that calls ”PI Ask for Communication Parameters.vi” first and then opens the specified communications ports. It does not, however, define the connected axes of the (motion) systems.

A more system-specific sample VI is “XXXX_Simple_Test.vi” (with XXXX being the PI product number of your system), also located in the “GCS_LabVIEW” main folder. It is available both for command-based and analog systems.

1.4. Working with two PI products which understand PI's General Command Set (GCS) in LabVIEW

When installing the LabVIEW programming support for two different PI products, there are two “Low Level” folders installed, one in each product-specific LabVIEW driver set. This is because every product comes with only the VIs which are used with the product. Another product may have different libraries or different library contents due to the product supporting more or fewer functions. When working with two product-specific LabVIEW driver set installations on one computer, it is important to make sure that LabVIEW always uses the right libraries.

a) When working separately with two products, the “Low Level” folder of each product must be located in the same folder as the product-specific main VI which calls sub-VIs from the product-specific driver set. Otherwise LabVIEW will start searching for sub VIs whereever it finds them, which may result in version conflicts and “broken Run” arrows. Please make sure that no VIs are saved under LabVIEWs own “user.lib” sub-folder. If they are LabVIEW will always find them there first, which will cause errors in many cases.

b) When working with two products in parallel, the libraries should be combined. Under LabVIEW 7.1 (but also under older LabVIEW versions), this is quite easy using LabVIEWs "Library manager", which can be found on the "Tools" menu. First, make a backup copy of the older library you want to combine with the newer one. In Library manager, open the newer library (e.g. "Special.llb" from the C-865 release) in the left window, and the older library (e.g. "Special.llb" from the older C-843 release) in the right window. Then choose "Show dates", "Disable files with identical dates", and "Sort alphabetically". This will show you only the VIs which have different file dates or which are only present in one of these libraries, but will not show any identical VIs. Now select all VIs which are still shown on the left window and copy them to the right window. In this way, you get a library with all VIs used by both product driver sets, and the newest version of each. Do this for all libraries in the low level folder. Make sure to work thereafter with the combined libraries instead of the product-specific libraries.

1.5. First Steps for GCS-Compatible PI Controllers

1.5.1. Analog systems

Step 1: To control an analog system, DAQmx 8.3 and a DAQmx compatible National Instruments DAC card must be installed. To use the patented HyperBit technology (U.S. Patent 6,950,050; international patents pending), which makes it possible to control an analog system with a physical resolution higher than the nominal output resolution of the DAC board, a password must be obtained from PI. Please contact your local sales representative for details about PIs HyperBit technology.

Step 2: Before powering up the controller/amplifier, make sure that any internal jumpers and switches are properly set. In particular, the control input voltage range must be set properly. See the respective User Manuals for details.

Step 3. With the controller still powered down, connect each positioner to the controller axis for which it was calibrated (usually marked on a label on the controller).

Step 4: With the controller still powered down, connect Analog IN (or Control IN) and Sensor Monitor lines from the controller axes to be under analog control to the DAC card in the host PC. If the controller does not have a Sensor Monitor Output, connect the Analog Out line of the DAC card to the corresponding Analog In line of the card.

Step 5: If the controller also has a command interface, and if commands are necessary to activate analog-input control, then those commands must be sent from any PC (could be the PC with the DAC card) using any software that supports the command interface (usually including LabVIEW). Most controllers are or can be configured with analog control as the power-up or factory default. See the controller User Manual and associated software manuals for details.

Step 6: Make sure all manual settings on the controller are correct. In particular, the servo state must be set as desired and the DC-Offset potentiometer, if present and not deactivated, should be set to 0.

Step 7: Power up the controller.

Step 8 (advanced users can skip this step): To check communication between the analog system and the host PC, run “Analog_Simple_Test.vi”. This VI will return the axis ID of one connected axis, its movement range and its current position value expressed in volts on a relative scale corresponding to the control input range. See Section 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 9:

WARNING: Analog_Configuration_Setup.vi May Cause Move

When you start “Analog_Configuration_Setup.vi” with All axes? = TRUE and Move to middle? = TRUE or Offset/Gain correction = TRUE, the VI will output voltages on the selected AO channels. It is therefore important to make sure that items connected to or mounted on stages connected to the corresponding analog channels of the controller cannot be damaged by such a move.

If an axis under analog control has a DC-offset knob on the controller or amplifier, make sure that it is in the zero-offset position (usually full counter-clockwise).

To begin controlling one or more analog channels with this driver set, run “Analog_Configuration_Setup.vi”. This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment:

- Definition of axis IDs,
- Definition of DAQmx input and output tasks,
- Determination of optimum AO voltage range and AO range offset for maximum physical bit resolution in given voltage range,
- Offset and gain correction (if appropriate) and
- Definition of the controller name.

During your testing phase (when you simply run the VIs without wiring them together into a program) do not close “Analog_Configuration_ Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work. The VI will stay idle when all tasks are finished and you can run all command VIs separately afterwards. Do not forget to press the STOP button when you have finished working manually with this driver set.

When programming your application, you should include “Analog_ Configuration_Setup.vi” as an initialization VI in your software. See chapter 3 for a detailed description of “Analog_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

When working with more than one axis and commanding a subset of them, the DAQmx task will always command all connected axes. The axes to command will be commanded to their given target positions and all other axes to their current positions, which may result in a small motion of the axes which were not commanded, due to drift . This occurs because a DAQmx task can only command all physical channels (i.e. axes) at once and not single channels (axes) separately. Voltages commanded or read with VOL.vi and VOL?.vi are the control voltages, not the resultant amplified and servo-controlled voltages seen by the piezos.

If you generate too many HyperBits by setting the PWM rate too high, the rate capacity of the card or the load capacity of the computer's bus can be exceeded. This may result in a runtime error, or sluggish operation during HyperBit analog output. Solutions: set your VIs to request a lower PWM rate; check your NIMax configuration settings to ensure DMA is being used; or use a faster card. Keep in mind that DAC granularity is only one of many possible limiters to system resolution. Ambient noise and vibration and electronic/amplifier noise, for example, also limit what your system can achieve. There is no point generating HyperBits below the fundamental noise floor.

If your controller has a digital display, values displayed on the display may differ slightly from values reported by LabVIEW because of the different measurement system, data conversion and software-internal offset- and gain corrections.

GCS syntax version: 2.0

1.5.2. C-702

Step 1: The C-702 controller is delivered pre-configured. Before you start, please check that the current configuration matches your stage connections. See the C-702 User Manual for a detailed description of this step.

Step 2 (advanced users can skip this step): To check communication between the C-702 controller and the host PC, run "C702_Simple_Test.vi". This VI will return the ID string of the C-702 controller, the axis IDs of the connected axes, and their current positions. See chapter 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions. Please make sure that you have all reported axes connected. If you want to work with only some of these axes, remember the IDs of those axes so that you can enter them in the Axes to move control in "C702_Configuration_ Setup.vi"; thereafter the other axes will not be moved when executing "C702_Configuration_Setup.vi".

Step 3:

WARNING: C702_Configuration_Setup.vi May Cause Move

When you start "C702_Configuration_Setup.vi" with All axes? = TRUE, the VI will automatically determine which axes have a reference switch and which have limit switches and, if the referencing mode of these axes is ON, will move these stages to these sensor switches. It is therefore important to make sure that items connected to or mounted on connected stages cannot be damaged by such a move. If referencing is not possible (because the connected stage has no reference or limit switch) or not desired, referencing mode (the mode which tells the controller to reference the stages or not) can be switched off (see chapter 3). See description of RON for details and warnings.

To control one or more C-702 controllers with this driver set, run "C702_Configuration_Setup.vi". This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of axis IDs, the initialization of the connected stages including referencing (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close "C702_Configuration_Setup.vi"; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement "C702_Configuration_Setup.vi" as an initialization VI in your software. See chapter 3 for a detailed description of "C702_Configuration_Setup.vi" and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Before using a joystick connected to the host PC, install joystick driver and calibrate joystick in the Windows control panel.

GCS syntax version: 1.0

1.5.3. C-843

To control one or more C-843 boards with this driver set, “C843_GCS_DLL.dll”, “MC.dll”, “PiStages.dat” and the C-843 device driver must be installed on your computer. See chapter 1.1 for information about methods for proper installation of the first three items. A description of how to install the C-843 device driver is given in the C-843 User Manual. The following steps must then be performed:

Step 1 (advanced users can skip this step): To check communication with the C-843 board in the host PC, run “C843_Simple_Test.vi”. This VI will return the ID string of the C-843 board and the axis IDs of the connected axes. See chapter 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 2:

WARNING: C843_Configuration_Setup.vi May Cause Move

When you start “C843_Configuration_Setup.vi” with All axes? = TRUE, the VI will automatically determine which axes have a reference switch and which have limit switches and, if the referencing mode of these axes is ON, will move these stages to these sensor switches. It is therefore important to make sure that items connected to or mounted on connected stages cannot be damaged by such a move. If referencing is not possible (because the connected stage has no reference or limit switch) or not desired, referencing mode (the mode which tells the controller to reference the stages or not) can be switched off. See description of RON for details and warnings.

Open “C843_Configuration_Setup.vi”. Select your C-843 board (2- or 4-axis version, board number) and leave “Use dialog to define connected stages” = TRUE. Run the VI. In the following screen, specify which stages you have connected to which axes and press OK. This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of axis IDs, the initialization of the connected stages including referencing (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “C843_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement “C843_Configuration_Setup.vi” as an initialization VI in your software. See chapter 3 for a detailed description of “C843_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

If you do not find your stage in the drop-down list, press CANCEL. You can then either define a User Stage with the Stage Editor, or you can contact PI to see about getting a new stage list: the "PIStages.dat" file contains all relevant stage parameters.

Default axis names are 1 to 4, but can be changed using “SAI.vi”.

GCS syntax version: 1.0

1.5.4. C-843.PM

With this driver set, “C843_PM_GCS_DLL.dll”, “MC.dll”, “PiStages.dat” and the C-843 device driver must be installed on your computer if you wish to control axes on PIs linear piezo motor stages. It can be used with axes on other motorized stages connected to the same or other C-843 boards in the same machine. See chapter 1.1 for information about methods for proper installation of the first three items. A description of how to install the C-843 device driver is given in the C-843 User Manual. The following steps must then be performed:

Step 1 (advanced users can skip this step): To check communication with the C-843 board in the host PC, run “C843_PM_Simple_Test.vi”. This VI will return the ID string of the C-843 board and the axis IDs of the connected axes. See chapter 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 2:

WARNING: C843_PM_Configuration_Setup.vi May Cause Move

When you start “C843_PM_Configuration_Setup.vi” with All axes? = TRUE, the VI will automatically determine which axes have a reference switch and which have limit switches and, if the referencing mode of these axes is ON, will move these stages to these sensor switches. It is therefore important to make sure that items connected to or mounted on connected stages cannot be damaged by such a move. If referencing is not possible (because the connected stage has no reference or limit switch) or not desired, referencing mode (the mode which tells the controller to reference the stages or not) can be switched off. See description of RON for details and warnings.

Open “C843_PM_Configuration_Setup.vi”. Select your C-843 board (2- or 4-axis version, board number) and leave “Use dialog to define connected stages” = TRUE. Run the VI. In the following screen, specify which stages you have connected to which axes and press OK. This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of axis IDs, the initialization of the connected stages including referencing (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “C843_PM_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement “C843_PM_Configuration_Setup.vi” as an initialization VI in your software. See chapter 3 for a detailed description of “C843_PM_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

If you do not find your stage in the drop-down list, press CANCEL. You can then either define a User Stage with the Stage Editor, or you can contact PI to see about getting a new stage list: the "PIStages.dat" file contains all relevant stage parameters.

Default axis names are 1 to 4, but can be changed using “SAI.vi”.

GCS syntax version: 1.0

1.5.5. C-848

Step 1: The C-848 controller is delivered pre-configured. Before you start, please check that the current configuration matches your stage connections. See the C-848 User Manual for a detailed description of this step.

Step 2 (advanced users can skip this step): To check communication between the C-848 controller and the host PC, run "C848_Simple_Test.vi". This VI will return the ID string of the C-848 controller, the axis IDs of the connected axes, and their current positions. See chapter 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions. Please make sure that you have all reported axes connected. If you want to work with only some of these axes, remember the IDs of those axes so that you can enter them in the Axes to move control in "C848_Configuration_ Setup.vi"; thereafter the other axes will not be moved when executing "C848_Configuration_Setup.vi".

Step 3:

WARNING: C848_Configuration_Setup.vi May Cause Move

When you start "C848_Configuration_Setup.vi" with All axes? = TRUE, the VI will automatically determine which axes have a reference switch and which have limit switches and, if the referencing mode of these axes is ON, will move these stages to these sensor switches. It is therefore important to make sure that items connected to or mounted on connected stages cannot be damaged by such a move. If referencing is not possible (because the connected stage has no reference or limit switch) or not desired, referencing mode (the mode which tells the controller to reference the stages or not) can be switched off (see chapter 3). See description of RON for details and warnings.

To control one or more C-848 controllers with this driver set, run "C848_Configuration_Setup.vi". This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of axis IDs, the initialization of the connected stages including referencing (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close "C848_Configuration_Setup.vi"; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement "C848_Configuration_Setup.vi" as an initialization VI in your software. See chapter 3 for a detailed description of "C848_Configuration_Setup.vi" and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Before using a joystick connected to the C-848 controller, calibrate the joystick by running "PI Terminal.vi". Type "JEN CALIB" and follow the instructions on the screen.

Before using a joystick connected to the host PC, install joystick driver and calibrate joystick in the Windows control panel.

GCS syntax version: 1.0

1.5.6. C-865

To control one or more C-865s with this driver set, “C865_GCS_DLL.dll”, “ MC_C865.dll ”, and “PiStages.dat” must be installed on your computer. See Section 1.1 for information about methods for proper installation of the first three items. The following steps must then be performed:

Step 1 (advanced users can skip this step): To check communication with the C-865 controller, run “C865_Simple_Test.vi”. This VI will return the ID string of the C-865 controller. See Section 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 2:

WARNING: C865_Configuration_Setup.vi May Cause Move

When you start “C865_Configuration_Setup.vi” with Is axis connected and can be moved? = TRUE, the VI will automatically determine if the connected axis has a reference switch or limit switch and, if the referencing mode of this axis is ON, will move the stage to one of the switches. It is therefore important to make sure that items connected to or mounted on the connected stage will not be damaged by such a move. If referencing is not possible (because the connected stage has no reference or limit switch) or not desired, referencing mode (the mode which tells the controller to reference the stage or not) can be switched off. See description of RON for details and warnings.

Open “C865_Configuration_Setup.vi”. Select the RS-232 settings (port number and appropriate baudrate) and leave Use dialog to define connected stage = TRUE. Run the VI. In the following screen, specify which stage you have connected and press OK. This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of the axis ID, the initialization of the connected stage, including referencing (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “C865_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement “C865_Configuration_Setup.vi” as an initialization VI in your software. See Section 3 for a detailed description of “C865_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

If you do not find your stage in the drop-down list, press CANCEL. You can then either define a User Stage with the Stage Editor, or you can contact PI to see about getting a new stage list: the "PIStages.dat" file contains all relevant stage parameters.

Default axis name is “1”, but can be changed using “SAI.vi”.

If the controller does not respond, please reset it using the reset button (unlabeled) on the rear panel of the C-865 controller.

See also “C865_Sample_Application_1.vi” and “C865_Sample_Application_1a.vi” as sample VIs showing how to implement “C865_Configuration_Setup.vi” as the initialization VI for the C-865 in your application.

Before using a joystick connected to the host PC, install joystick driver and calibrate joystick in the Windows control panel.

GCS syntax version: 1.0

1.5.7. C-866

To control one or more C-866s with this driver set, “C866_GCS_DLL.dll”, “MC_C866.dll” and “PiStages.dat” must be installed on your computer. See Section 1.1 for information about methods for proper installation of the first three items. The following steps must then be performed:

Step 1 (advanced users can skip this step): To check communication with the C-866 controller, run “C866_Simple_Test.vi”. This VI will return the ID string of the C-866 controller. See Section 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 2:

WARNING: C866_Configuration_Setup.vi May Cause Move

When you start “C866_Configuration_Setup.vi” with Is axis connected and can be moved? = TRUE, the VI will automatically determine if the connected axis has a reference switch or limit switch and, if the referencing mode of this axis is ON, will move the stage to one of the switches. It is therefore important to make sure that items connected to or mounted on the connected stage will not be damaged by such a move. If referencing is not possible (because the connected stage has no reference or limit switch) or not desired, referencing mode (the mode which tells the controller to reference the stage or not) can be switched off. See description of RON for details and warnings.

Open “C866_Configuration_Setup.vi”. Select the correct interface settings and leave Use dialog to define connected stage = TRUE. Run the VI. In the following screen, specify which stage you have connected and press OK. This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of the axis ID, the initialization of the connected stage, including referencing (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “C866_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement “C866_Configuration_Setup.vi” as an initialization VI in your software. See Section 3 for a detailed description of “C866_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions. If you do not find your stage in the drop-down list, press CANCEL. You can then either define a User Stage with the Stage Editor, or you can download a newer version of the stage list file "PIStages.dat" (which contains all relevant stage parameters) from our homepage www.pi.ws (see C-866 User Manual for a download instruction).

Default axis name is “1”, but can be changed using “SAI.vi”.

If the controller does not respond, please reset it using the recessed Reset button on the front panel.

See also “C866_Sample_Application_1.vi”, “C866_Sample_Application_1a.vi” and “C866_Sample_Application_2.vi” as sample VIs showing how to implement “C866_Configuration_Setup.vi” as the initialization VI for the C-866 in your application.

Before using a joystick connected to the host PC, install joystick driver and calibrate joystick in the Windows control panel.

GCS syntax version: 1.0

1.5.8. C-880

Step 1: The C-880 controller is delivered pre-configured. Before you start, please check that the current configuration matches your stage connections. See the C-880 User Manual for a detailed description of this step.

Step 2 (advanced users can skip this step): To check communication between the C-880 controller and the host PC, run “C880_Simple_Test.vi”. This VI will return the ID string of the C-880 controller, the axis IDs of the connected axes, and their current positions. See chapter 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions. Please make sure that you have all reported axes connected. If you want to work with only some of these axes, remember the IDs of those axes so that you can enter them in the Axes to move control in “C880_Configuration_ Setup.vi”; thereafter the other axes will not be moved when executing “C880_Configuration_Setup.vi”.

Step 3:

WARNING: C880_Configuration_Setup.vi May Cause Move

When you start “C880_Configuration_Setup.vi” with All axes? = TRUE, the VI will automatically determine which axes have a reference switch and which have limit switches and, if the referencing mode of these axes is ON, will move these stages to these sensor switches. It is therefore important to make sure that items connected to or mounted on connected stages cannot be damaged by such a move. If referencing is not possible (because the connected stage has no reference or limit switch) or not desired, referencing mode (the mode which tells the controller to reference the stages or not) can be switched off (see chapter 3). See description of RON for details and warnings.

To control one or more C-880 controllers with this driver set, run “C880_Configuration_Setup.vi”. This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of axis IDs, the initialization of the connected stages including referencing (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “C880_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement “C880_Configuration_Setup.vi” as an initialization VI in your software. See chapter 3 for a detailed description of “C880_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Before using a joystick connected to the C-880 controller, calibrate the joystick by running “PI Terminal.vi”. Type “JEN CALIB” and follow the instructions on the screen.

Before using a joystick connected to the host PC, install joystick driver and calibrate joystick in the Windows control panel.

GCS syntax version: 1.0

1.5.9. C-880K005

To control a C-880K005 controller with this driver set, you must perform the following steps:

1. Run “PI Open Interface.vi”
2. Run “Multix axis assignment.vi”.
3. Reference all connected stages using “PI_Multix.vi” with Command = REF and All axes? = TRUE. If no referencing is desired, refencing mode can be switched off using Command = RON. With referencing mode off, only relative moves can be commanded (using Command = MVR), unless the actual position is set with Command = POS.vi. Thereafter both relative and absolute moves can be commanded.

OR

1. Run C880K005_Simple_Test.vi
2. Proceed as in step 3 above.

After these steps, a number of selected commands can be used by calling “PI_Multix.vi”.

The control concept of the C-880K005 is based on the assumption that several C-880 multi-axis motion controllers are connected to the C-880K005, which is commanded over a single interface from the host PC. In this way, the number of axes to command is not limited by the number of connectors available on a single C-880. To ease handling so many axes, the user does not have to worry about the individual C-880 controllers, but only the C-880K005 controller with axes 1 to N, N being the sum of all axes connected to all interconnected C-880s.

The C-880K005 is called the “controller” and handles the communication to all connected C-880 controllers, which are called VControllers (virtual controllers). “PI_Multix.vi” must be used to send commands. For ease of operation, when running “Multix axis assignment.vi”, all connected axes of all connected C-880 controllers are queried and the axis IDs 1 to N are assigned to these axes automatically.

A C-880 connected to the C-880K005 can be directly commanded by setting it active (“ACT.vi”). The C-880K005 communications controller then passes subsequent commands to the active C-880, except for commands which, by their nature, must be directly handled by the C-880K005 (e.g. WAA).

Example: Three C-880 controllers are connected to the C-880K005, and each C-880 controller has 12 axes designated A to L on each of the C-880’s. These ID’s are taken as VAxis IDs and the axis IDs 1 to 36 are assigned for the C-880K005 controller. To command axis A of C-880 1 (VController 1) and axis B of C-880 3 (VController 3), the user commands axes 1 and 26 in “PI_Multix.vi”; the axis and value parsing is done internally.

See “C880K005_Simple_Test.vi” and “C880K005_Configuration_Setup.vi” for sample programs for the driver configuration of the C-880K005.

1.5.10. E-516

Step 1 (advanced users can skip this step): To check communication with the E-516 controller, run “E516_Simple_Test.vi”. This VI will return the ID string and the help string of the E-516 controller, the available axis IDs and positions of all axes. See Section 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 2:

WARNING: E516_Configuration_Setup.vi May Cause Move

When you start “E516_Configuration_Setup.vi” with Move all axes to middle? = TRUE, or Move all axes to middle? = FALSE and Axes to move = TRUE for some axes, the VI will move all axes or the selected axes to their middle positions. It is therefore important to make sure that items connected to or mounted on the connected stages will not be damaged by such a move.

Open “E516_Configuration_Setup.vi”. First select the interface settings (Interface = “RS232” or “GPIB”, RS232 settings = Portnumber and appropriate Baudrate, or GPIB settings = Bus and Address). Select whether a wave generator output is to be stopped (if you are not sure if there is any wave generator output running, leave this control TRUE) and whether the axes are to be moved to the midpoints of their travel ranges.

Then run the VI. It performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: definition of the axis IDs, the online setting of the controller, the servo setting of the axes and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “E516_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can include “E516_Configuration_Setup.vi” as an initialization VI in your software. See Section 3 for a detailed description of “E516_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Default position unit is µm, default velocity unit is µm/ms.

To use wave-generator-specific VI’s, whose names start with “WGWaveEditor_*.vi”, “WGWaveEditor.dll” must be installed on your computer. During the installation of “WGWaveEditor.dll” “NTGraph.ocx” will be installed also. If “NTGraph.ocx” is not registered correctly in the Windows environment, the editor of “WGWaveEditor.dll” will not function.

See “E516_WaveGenerator_Sample_Program.vi” for a sample program using these VIs.

GCS syntax version: 1.0

1.5.11. E-710

To control one or more E-710s with this driver set, “E7XX_GCS_DLL.dll” must be installed on your computer. See Section 1.1 for information about methods for a proper installation. The following steps must then be performed:

Step 1 (advanced users can skip this step): To check communication with the E-710 controller, run “E710_Simple_Test.vi”. This VI will return the ID string of the E-710 controller and the available axis IDs. See Section 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display a *Context Help* window with the control/indicator descriptions.

Step 2:

WARNING: E710_Configuration_Setup.vi May Cause Move

When you start “E710_Configuration_Setup.vi” with Perform autozero? = TRUE and/or Move to middle? = TRUE, the VI will perform an automated zero-point calibration of the connected linear axes and/or move these axes to their middle positions. It is therefore important to make sure that items connected to or mounted on the connected stages will not be damaged by such a move.

Open “E710_Configuration_Setup.vi”. First select the interface settings (Interface = “RS232” or “GPIB”, RS232 settings = Portnumber and appropriate Baudrate, or GPIB settings = Bus and Address), and specify if stages are connected. Select whether the automated zero-point calibration is to be performed (linear axes only), whether the Low voltage parameter is to be defined automatically or manually, and whether the axes are to be moved to the midpoints of their travel ranges.

WARNING:

The repeat write times of the internal non-volatile memory of the E-710 controller are limited. Do not use commands which write to the EEPROM (e.g. WPA, SEP) except when necessary. See User Manual for details.

Then run the VI. It performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of the axis IDs, the initialization of the axes (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “E710_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement “E710_Configuration_Setup.vi” as an initialization VI in your software. See Section 3 for a detailed description of “E710_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Default axis IDs are “1”,”2”,”3”,”4”.
Default position unit is µm, default velocity unit is µm/ms.

Due to the emulation of the native E-710 command set, the execution of “MOV.vi” and “MVR.vi” is noticeably slower than that of the native firmware commands. Therefore this driver set provides the special non-GCS motion functions “NMOV.vi” and “NMVR.vi” for the case that your application requires quickest possible response to motion commands. See the VI reference of these two VIs for details.

Before using a joystick connected to the host PC, install joystick driver and calibrate joystick in the Windows Control Panel.

GCS syntax version: 1.0

1.5.12. E-753

Step 1: The E-753 controller is delivered pre-configured. Before you start, please check that the current configuration matches your stage connections.

Step 2 (advanced users can skip this step): To check communication between the E-753 controller and the host PC, run “E753_Simple_Test.vi”. This VI will return the ID string of the E-753 controller, the axis ID and stage name of the connected axis, and its current position. See chapter 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 3:

WARNING: E753_Configuration_Setup.vi May Cause Move

When you start “E753_Configuration_Setup.vi” with Connected? = TRUE and Perform Autozero? = TRUE or Move to Middle? = TRUE, the VI will move the stage. It is therefore important to make sure that items connected to or mounted on the connected stage cannot be damaged by such a move. If a move is not desired, Connected? can be switched off (see chapter 3).

To control one or more E-753 controllers with this driver set, run “E753_Configuration_Setup.vi”. This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of the axis ID, the automated zero-point calibration (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “E753_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work.

When programming your application, you can implement “E753_Configuration_Setup.vi” as an initialization VI in your software. See chapter 3 for a detailed description of “E753_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions. See also “E753_Sample_Application_1.vi” and “E753_Sample_Application_2a.vi” as sample VIs showing how to implement your application using “E753_Configuration_Setup.vi”.

Before using a joystick connected to the host PC, install joystick driver and calibrate joystick in the Windows control panel.

Default position unit is µm, default velocity unit is µm/s.

GCS syntax version: 2.0

1.5.13. E-755

Step 1: The E-755 controller is delivered pre-configured. Before you start, please check that the current configuration matches your stage connections.

Step 2 (advanced users can skip this step): To check communication between the E-755 controller and the host PC, run “E755_Simple_Test.vi”. This VI will return the ID string of the E-755 controller, the axis ID of the connected axis, and its current position/voltage (depending on closed-loop or open-loop controller). See chapter 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 3:

WARNING: E755_Configuration_Setup.vi May Cause Move

When you start “E755_Configuration_Setup.vi” with <u>Connected?</u> = TRUE, the VI will automatically determine if the axis can be referenced and, if it can, will move the stage to the positive or negative hard stop. It is therefore important to make sure that items connected to or mounted on the connected stage cannot be damaged by such a move. If referencing is not possible (because the controller is an open-loop device, or the hard stop of the connected stage cannot be used for referencing) or not desired, <u>Connected?</u> can be switched off (see chapter 3).

To control one or more E-755 controllers with this driver set, run “E755_Configuration_Setup.vi”. This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of axis ID, the initialization of the connected stage including referencing (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “E755_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work.

Do not forget to run “Close connection if open.vi” with <u>Close DaisyChain</u> = TRUE before re-connecting this or any other controller connected to the same interface (except if you want to connect another device to the same DaisyChain).

When programming your application, you can implement “E755_Configuration_Setup.vi” as an initialization VI in your software. See chapter 3 for a detailed description of “E755_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Before using a joystick connected to the host PC, install joystick driver and calibrate joystick in the Windows control panel.

Default position unit is µm, default velocity unit is µm/s.

GCS syntax version: 2.0

1.5.14. E-761

To control one or more E-761 boards with this driver set, “E7XX_GCS_DLL.dll” must be installed on your computer. See Section 1.1 for information about methods for a proper installation. The following steps must then be performed:

Step 1 (advanced users can skip this step): To check communication with the E-761 controller, run “E761_Simple_Test.vi”. This VI will return the ID string of the E-761 controller and the available axis IDs. See Section 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 2:

WARNING: E761_Configuration_Setup.vi May Cause Move

When you start “E761_Configuration_Setup.vi” with Perform autozero? = TRUE and/or Move to middle? = TRUE, the VI will perform an automated zero-point calibration of the connected linear axes and/or move these axes to their middle positions. It is therefore important to make sure that items connected to or mounted on the connected stages will not be damaged by such a move.

Open “E761_Configuration_Setup.vi”. First select the Board number and specify if stages are connected. Select whether the automated zero-point calibration is to be performed (linear axes only), whether the Low voltage parameter is to be defined automatically or manually, whether the servo status is to be changed to ON, whether a wave generator output is to be stopped (if you are not sure if there is any wave generator output running, leave this control TRUE because AutoZero cannot be performed while a wave generator is running) and whether the axes are to be moved to the midpoints of their travel ranges.

Then run the VI. It performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of the axis IDs, the initialization of the axes (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “E761_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement “E761_Configuration_Setup.vi” as an initialization VI in your software. See Section 3 for a detailed description of “E761_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Default axis IDs are “1”,”2”,”3”. Default piezo channel IDs are “1”, “2”, “3”, “4”. Defauls sensor IDs are “1”, “2”, “3”. Default analog input board no. is “4”.

Default position unit is µm, default velocity unit is µm/ms.

Before using a joystick connected to the host PC, install joystick driver and calibrate joystick in the Windows Control Panel.

GCS syntax version: 1.0

1.5.15. E-816

When controlling the E-816, timing problems can occur if several command VIs are run in rapid sequence, resulting in lost commands. To prevent such communication errors, it is recommended that you include a certain wait time between the different programming steps, depending on the command to be executed. This is especially true for commands that need a certain execution time inside the E-816 module, like MOV, MVR, SPA, SVA, SVR, RST, WPA, SWT and WTO. Only one axis per command can be controlled. “Split num query command.vi” can be used to query POS?, MOV?, VOL?, SVA? for multiple axes at a time.

GCS syntax version: 1.0

1.5.16. F-206

This driver set (PI General LabVIEW Driver Set) and the F-206 LabVIEW driver set that also comes with the F-206 system are fully compatible and can be used in parallel. The F-206 can be fully controlled with the PI General LabVIEW Driver Set.The axis identifiers of the F-206 (X,Y,Z,U,V,W), NanoCube (K,L,M, if present) and additional axes (A,B, if any) cannot be changed.

Step 1 (advanced users can skip this step): To check communication between the F-206 controller and the host PC, run "F206_Simple_Test.vi". This VI will return the ID and help strings of the F-206 controller and the axis IDs and stage names of the connected axes (according to your selection of Is a NanoCube present? and How many additional axes are present?). If you have ordered the AC8 option, you can drive up to two additional separate, motor-driven axes (PWM-compatible motors with position control) with the F-206 controller (see also the F-206 User Manual). If you have ordered the NCU option, you can drive a 3-axis piezo stage ("NanoCube") with the F-206 controller. Before you proceed with step 2, please check that the current configuration matches your stage connections. See chapter 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 2:

WARNING: F206_Configuration_Setup.vi May Cause Move

When you start "F206_Configuration_Setup.vi" with Initialize hexapod? = TRUE and/or Initialize additional axes? = TRUE, the VI will automatically move the Hexapod (and NanoCube, if present) and/or the additional axes to their sensor switches. It is therefore important to make sure that items connected to or mounted on connected stages cannot be damaged by such a move.

To control one or more F-206 controllers with this driver set, run "F206_Configuration_Setup.vi". This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of axis IDs, the initialization of the connected stages including referencing (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close "F206_Configuration_Setup.vi"; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement "F206_Configuration_Setup.vi" as an initialization VI in your software. See chapter 3 for a detailed description of "F206_Configuration_Setup.vi" and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

GCS syntax version: 1.0

1.5.17. M-840 / M-850

This driver set (PI General LabVIEW Driver Set) and the M-840 / M-850 LabVIEW driver set that comes with the M-840 / M-850 system are fully compatible and can be used in parallel. The M-840 / M-850 can be fully controlled with the PI General LabVIEW Driver Set and is called “M-8X0” from here on. The axis identifiers of the M-840 / M-850 and additional axes (if any) cannot be changed.

Step 1 (advanced users can skip this step): To check communication between the M-8X0 controller and the host PC, run “M8X0_Simple_Test.vi”. This VI will return the ID and help strings of the M-8X0 controller and the axis IDs and stage names of the connected axes (according to your selection of How many additional axes? are connected to the M-8X0 controller). If you have ordered the AC8 option, you can drive up to two additional separate, motor-driven axes (PWM-compatible motors with position control) with the M-8X0 controller (see also the M-8X0 User Manual). Before you proceed with step 2, please check that the current configuration matches your stage connections. See chapter 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 2:

WARNING: M8X0_Configuration_Setup.vi May Cause Move

When you start “M_8X0_Configuration_Setup.vi” with Initialize hexapod? = TRUE and/or Initialize additional axes? = TRUE, the VI will automatically move the Hexapod and/or the additional axes to their sensor switches. It is therefore important to make sure that items connected to or mounted on connected stages cannot be damaged by such a move.

To control one or more M-8X0 controllers with this driver set, run “M8X0_Configuration_Setup.vi”. This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of axis IDs, the initialization of the connected stages including referencing (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “M8X0_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement “M8X0_Configuration_Setup.vi” as an initialization VI in your software. See chapter 3 for a detailed description of “M8X0_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

GCS syntax version: 1.0

1.5.18. Mercury™

To control one or more Mercury™ controller with this driver set, “Mercury_GCS_DLL.dll” and “PiStages.dat” must be installed on your computer. See Section 1.1 for information about methods for proper installation of these items. The following steps must then be performed:

Step 1 (advanced users can skip this step): To check communication with the Mercury™ controller, run “Mercury_Simple_Test.vi”. This VI will return the ID string of the Mercury™ controller. See Section 3 for a description of this VI and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Step 2:

WARNING: Mercury_Configuration_Setup.vi May Cause Move

When you start “Mercury_Configuration_Setup.vi” with Are all axes connected and can be moved? = TRUE, the VI will automatically determine if the connected axes have a reference switch or limit switch and, if the referencing mode of these axes is ON, will move the stages to one of the switches. It is therefore important to make sure that items connected to or mounted on the connected stages will not be damaged by such a move. If referencing is not possible (because the connected stage has no reference or limit switch) or not desired, referencing mode (the mode which tells the controller to reference the stage or not) can be switched off. See description of RON for details and warnings.

Open “Mercury_Configuration_Setup.vi”. Select the RS-232 settings (port number and appropriate baudrate) and leave Use dialog to define connected stages = TRUE. Run the VI. In the following screen, specify which stages you have connected and press OK. This VI performs all steps necessary for a full configuration of the driver VIs in the LabVIEW environment: the definition of the axis ID, the initialization of the connected stages, including referencing (if appropriate) and the definition of the controller name. During your testing phase (when you simply run the VIs without wiring them together into a program), do not close “Mercury_Configuration_Setup.vi”; otherwise all global settings will be lost and the driver VIs will not work. When programming your application, you can implement “Mercury_Configuration_Setup.vi” as an initialization VI in your software. See Section 3 for a detailed description of “Mercury_Configuration_Setup.vi” and use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

If you do not find your stage in the drop-down list, press CANCEL. You can then either define a User Stage with the Stage Editor, or you can contact PI to see about getting a new stage list: the "PIStages.dat" file contains all relevant stage parameters.

The default axis names are determined by the device address (see Mercury™ User Manual), typically “A” – “P”, and can be changed using “SAI.vi”.

See also “Mercury_Sample_Application_1.vi” and “Mercury_Sample_Application_1a.vi” as sample VIs showing how to implement “Mercury_Configuration_Setup.vi” as the initialization VI for the Mercury in your application.

GCS syntax version: 1.0

2. Low Level VIs

The following low-level VIs can be found in the “Low Level” folder:

2.1. Communication VIs (“Communication.llb”):

2.1.1. Available DLL interfaces.ctl

Valid for	C-843, C-843.PM, C-865, C-866, E-710, E-755, E-761, Mercury™ (but must be present for all other systems also)
Input	None
Output	None
Remarks	Type definition for hardware interfaces available when communicating with a system through a PI GCS DLL.

2.1.2. Available DLLs.ctl

Valid for	C-843, C-843.PM, C-865, C-866, E-710, E-755, E-761, Mercury™ (but must be present for all other systems also)
Input	None
Output	None
Remarks	Type definition for available GCS DLLs for communicating with a system.

2.1.3. Available interfaces.ctl

Valid for	All systems
Input	None
Output	None
Remarks	Type definition for available interfaces for communicating with a system.

2.1.4. Close connection if open.vi

Valid for	All systems
Input	System number (1), Error in (no error)
Output	Was connected? (T/F), Error out
Remarks	This VI checks if the connection to the selected system is already open and, if it is, it closes this connection.

2.1.5. GCSTranslator DLL Functions.vi

Valid for	C-843, C-843.PM, C-844, C-865, C-866, E-710, E-755, E-761, Mercury™ (but must be present in Communication.llb for all other systems also)
Input	System number (1), Function (C844_IsDLLAvailable), String buffer (empty string), String input (empty string), Error in (no error)
Output	DLL I32 Return value, Numerical output, Boolean output (T/F), String output, Error out
Remarks	This VI calls a given function from GCSTranslator.dll. GCSTranslator.dll

must be installed. To call a system-specific function, the system-specific GCS DLL must be installed also.

Warning: For XXX_GcsGetANswer , String buffer must be large enough, otherwise the application may crash. Call XXX_GcsGetANswerSize first to determine necessary string length.

2.1.6. Global DaisyChain.vi

Valid for	All systems
Input	None
Output	None
Remarks	A global variable which contains setup information for DaisyChain systems.

2.1.7. Global1.vi

Valid for	All systems
Input	None
Output	None
Remarks	A global variable which contains communication setup information.

2.1.8. Initialize Global1.vi

Valid for	All systems
Input	System number (1), Error in (no error)
Output	Error out
Remarks	This VI initializes Global1 according to the given system no.

2.1.9. PI Receive String.vi

Valid for	All systems
Input	System number (1), Strip spaces? (F), Error in (no error)
Output	String read, Bytes read, Error out
Remarks	Read string from selected system.

2.1.10. PI Send String.vi

Valid for	All systems
Input	System number (1), String to send (empty string), Attach termination char.? (T), Error in (no error)
Output	Error out
Remarks	Sends command with or without trailing termination character to selected system.

2.1.11. PI VISA Receive Characters.vi

Valid for	C-702, C-848, C-880, C-880K005, E-516, E-753, E-816, F-206, M-8X0 (but must be present in Communication.llb for all other systems also)
Input	System number (1), Bytes to read (1), Error in (no error)
Output	String read, Bytes read, Error out
Remarks	This vi reads n bytes (characters) via the chosen VISA interface. Sub-vi for "PI Receive String.vi".

2.1.12. Syntax.ctl

Valid for	All systems
Input	None
Output	None
Remarks	Type definition for GCS version.

2.1.13. Termination character.ctl

Valid for	All systems
Input	None
Output	None
Remarks	Type definition for termination character.

2.2. General Command VIs (“General command.llb”):

2.2.1. *IDN?.vi

Valid for	All systems
Input	System number (1), Error in (no error)
Output	Identification, Error out
Remarks	Returns system identification string.

2.2.2. Controller names.ctl

Valid for	All systems
Input	None
Output	None
Remarks	Type definition for control Controller names.

2.2.3. Define connected axes.vi

Valid for	All systems
Input	System number (1), Read from controller?(F), Invert order?(F), Conn. axes (empty string array), Error in (no error)

	Analog: Only supported when called by Analog_Configuration_Setup.vi
	C-702: Read from controller = TRUE, Invert order = TRUE
	C-843: Read from controller = TRUE, Invert order = FALSE
	C-843.PM: Read from controller = TRUE, Invert order = FALSE
	C-844: Read from controller = TRUE, Invert order = FALSE
	C-848: Read from controller = TRUE, Invert order = TRUE
	C-865: Read from controller = TRUE, Invert order = FALSE
	C-866: Read from controller = TRUE, Invert order = FALSE
	C-880: Read from controller = TRUE, Invert order = TRUE
	E-516: Read from controller = TRUE, Invert order = FALSE
	E-710: Read from controller = TRUE, Invert order = FALSE
	E-753: Read from controller = TRUE, Invert order = FALSE
	E-755: Read from controller = TRUE, Invert order = FALSE
	E-761: Read from controller = TRUE, Invert order = FALSE
	E-816: Read from controller = TRUE, Invert order = FALSE
	F-206: Read from controller = FALSE, Invert order = FALSE, Connected axes = X,Y,Z,U,V,W, (A,B,K,L,M optional)
	M-8X0: Read from controller = FALSE, Invert order = FALSE, Connected axes = X,Y,Z,U,V,W, (A,B optional)
	Mercury™: Read from controller = TRUE, Invert order = FALSE
Output	Connected axes out, Error out
Remarks	Writes connected axes into Global2 (Array).vi. **This VI is called automatically by “XXXX_Configuration_Setup.vi” (with XXXX being the PI product number of your system) and must be completed successfully before any other axis-specific command VI is called.** Requires “SAI?.vi” to be present.

2.2.4. Define connected systems (Array).vi

Valid for	All systems
Input	Controller names (array of Enum controls, none), Change only one system? (F), System number (1), Error in (no error)
	Analog system: Only supported when called by Analog_Configuration_Setup.vi
Output	Controller names out, Error out
Remarks	Defines connected systems and writes controller names into Global2 (Array).vi. **This VI is called automatically by “XXXX_Configuration_Setup.vi” (with XXXX being the PI product number of your system) and must be completed successfully before“General wait for movement to stop.vi” is called.** If Change only one system? is FALSE, all entries from Controller names are written into “Global2 (Array).vi”. If Change only one system? is TRUE, only the entry for the given system number is overwritten in “Global2 (Array).vi”.

2.2.5. ERR?.vi

Valid for	All systems.
Input	System number (1), Error in (no error)
Output	Controller error (T/F), Error out Analog: VI does not report any errors.
Remarks	Returns error information. Controller error is TRUE if selected system reports error code ≠ 0. See appendix A of this manual for a list of PI error codes and use “GCSTranslateError.vi” to translate error codes into error descriptions programmatically.

2.2.6. Global2 (Array).vi

Valid for	All systems
Input	System (array of Conn. axes (empty string array), Controller name (Enum control, none))
Output	None
Remarks	A global variable which contains identifiers for all connected axes of all connected systems and the names of all connected systems.

2.2.7. HLP?.vi

Valid for	Analog systems, C-702, C-843, C-843.PM, C-848, C-865, C-866, C-880, E-516, E-710, E-753, E-755, E-761, Mercury™
Input	System number (1), Error in (no error)
Output	Help string, Error out
Remarks	Returns help string.

2.2.8. Initialize Global2.vi

Valid for	All systems
Input	System number (1), Error in (no error)
Output	Error out
Remarks	This VI initializes Global2 (Array) according to the given system no.

2.2.9. MOV.vi

Valid for	Analog systems, C-702, C-843, C-843.PM, C-844, C-848, C-865, C-866, C-880, C-880K005, E-516, E-710, E-753, E-755, E-761, E-816, F-206, M-8X0, Mercury™
Input	System number (1), Axes to move (empty string array), Position values (empty num. array, 0), No. of digits (4), Error in (no error) C-880K005: VI only supported when called through PI_Multix.vi E-753: Motion commands are not allowed when the wave generator is active or the analog input is used for target generation. E-755: Command not available for E-755.101. E-816: Only one axis per command allowed. It is necessary to wait a certain time before sending the next command to prevent it from being lost. F-206: No mix between F-206 axes X,Y,Z,U,V,W and separate axes A,B allowed

Output	Error out
Remarks	Moves specified axes to specified absolute positions. No. of digits is the number of digits after the decimal point in the position value(s) that will be sent. E-710: See also “NMOV.vi” in “Old commands.llb”.

2.2.10. MVR.vi

Valid for	Analog systems, C-702, C-843, C-843.PM, C-844, C-848, C-865, C-866, C-880, C-880K005, E-516, E-710, E-753, E-755, E-761, E-816, Mercury™
Input	System number (1), Axes to move (empty string array), Position values (empty num. array, 0), No. of digits (4), Error in (no error) C-880K005: VI only supported when called through PI_Multix.vi E-755: Command not available for E-755.101. E-816: Only one axis per command allowed. It is necessary to wait a certain time before sending the next command to prevent it from being lost.
Output	Error out
Remarks	Moves specified axes **relative** to current position. No. of digits is the number of digits after the decimal point in the position value(s) that will be sent. E-710: See also “NMVR.vi” in “Old commands.llb”. E-753: Motion commands are not allowed when the wave generator is active or the analog input is used for target generation.

2.2.11. ONT?.vi

Valid for	C-702, C-843, C-843.PM, C-848, C-865, C-866, C-880, E-516, E-710, E-753, E-755, E-761, E-816, Mercury™ (but must be present for all other systems also)
Input	System number (1), Axes to query (empty string array), All axes? (F), Axis identifier? (T), Error in (no error) C-702: If All axes? = TRUE, then Axis identifier? must be TRUE C-843: If All axes? = TRUE, then Axis identifier? must be TRUE C-843.PM: If All axes? = TRUE, then Axis identifier? must be TRUE C-848: If All axes? = TRUE, then Axis identifier? must be TRUE C-865: If All axes? = TRUE, then Axis identifier? must be TRUE C-866: If All axes? = TRUE, then Axis identifier? must be TRUE C-880: If All axes? = TRUE, then Axis identifier? must be TRUE E-516: If All axes? = TRUE, then Axis identifier? can be FALSE. E-710: If All axes? = TRUE, then Axis identifier? can be FALSE. E-753: If All axes? = TRUE, then Axis identifier? can be FALSE. E-755: If All axes? = TRUE, then Axis identifier? can be FALSE. Command not available for E-755.101 E-761: If All axes? = TRUE, then Axis identifier? can be FALSE. E-816: All axes? = FALSE, only one axis per command allowed. Mercury™: If All axes? = TRUE, then Axis identifier? can be FALSE.

Output	Axis on target? (T/F), Error out
Remarks	Indicates whether or not queried axis is at target position.

2.2.12. POS?.vi

Valid for: Analog systems, C-702, C-843, C-843.PM, C-844, C-848, C-865, C-866, C-880, C-880K005, E-516, E-710, E-753, E-755, E-761, E-816, F-206, M-8X0, Mercury™

Input: System number (1), Axes to query (empty string array), All axes? (F), Axis identifier? (T), Error in (no error)

Analog: If All axes? = TRUE, then Axis identifier? can be FALSE.

C-702: If All axes? = TRUE, then Axis identifier? can be FALSE

C-843: If All axes?= TRUE, then Axis identifier? can be FALSE

C-843.PM: If All axes? = TRUE, then Axis identifier? can be FALSE

C-844: If All axes? = TRUE, then Axis identifier? can be FALSE

C-848: If All axes? = TRUE, then Axis identifier? can be FALSE

C-865: If All axes= TRUE, then Axis identifier? can be FALSE

C-866: If All axes= TRUE, then Axis identifier? can be FALSE

C-880: If All axes? = TRUE, then Axis identifier? can be FALSE

C-880K005: VI only supported when called through PI_Multix.vi

E-516: If All axes? = TRUE, then Axis identifier? must be TRUE

E-710: If All axes? = TRUE, then Axis identifier? must be TRUE

E-753: If All axes= TRUE, then Axis identifier? can be FALSE

E-755: If All axes? = TRUE, then Axis identifier? can be FALSE. Command not available for E-755.101.

E-761: If All axes? = TRUE, then Axis identifier? can be FALSE

E-816: All axes? = FALSE, only one axis per command allowed.

F-206: If All axes? = TRUE, then Axis identifier? must be TRUE

M-8X0: If All axes? = TRUE, then Axis identifier? must be TRUE

Mercury™: If All axes?= TRUE, then Axis identifier? can be FALSE

Output: Position, Error out

Remarks: Returns position information (actual or target position, depending on system).

F-206: Returned position value is the commanded target position for the axis.

M-8X0: Returned position value is the commanded target position for the axis.

2.2.13. SAI?.vi

Valid for: Analog systems, C-702, C-843, C-843.PM, C-844, C-848, C-865, C-866, C-880, C-880K005, E-516, E-710, E-753, E-755, E-761, E-816, Mercury™ (but must be present in “General command.llb” for all other systems also)

Input: System number (1), Invert order? (F), SAI? ALL (F), Error in (no error)

Analog: Invert order should be FALSE, SAI? ALL must be FALSE

C-702: Invert order should be TRUE, SAI? ALL must be FALSE

C-843: Invert order should be FALSE, SAI? ALL is supported

C-843.PM: Invert order should be FALSE, SAI? ALL must be FALSE

C-844: Invert order should be FALSE, SAI? ALL must be FALSE

C-848: Invert order should be TRUE, SAI? ALL must be FALSE

C-865: Invert order should be FALSE, SAI? ALL is supported

C-866: Invert order should be FALSE, SAI? ALL is supported

C-880: Invert order should be TRUE, SAI? ALL must be FALSE to read all configured axis IDs and must be TRUE to get all physically defined axis IDs

C-880K005: VI only supported when called through PI_Multix.vi, SAI? ALL must be FALSE

E-516: Invert order should be FALSE, SAI? ALL must be FALSE

E-710: Invert order should be FALSE, SAI? ALL is supported

E-753: Invert order should be FALSE, SAI? ALL is supported

E-755: Invert order should be FALSE, SAI? ALL is supported

E-761: Invert order should be FALSE, SAI? ALL is supported

E-816: Invert order should be FALSE, SAI? ALL must be FALSE

Mercury™: Invert order should be FALSE, SAI? ALL is supported

Output	Connected axes, Error out
Remarks	Returns axis identifiers of all configured axes and writes them into Global2 (Array).vi. If SAI? ALL is TRUE, all physically available axes are returned, no matter if configured or not. Required by “Define connected axes.vi”

2.2.14. STP.vi

Valid for	Analog systems, C-702, C-843, C-843.PM, C-844, C-848, C-865, C-866, C-880, E-516, E-753, E755, E-761, Mercury™ (but must be present for E-710 also)
Input	System number (1), Affected axes? (empty string array), All axes? (F), Axis identifier? (T), Error in (no error)

Analog: All axes? = TRUE, Axis identifier = FALSE. STP does not set any error code.

C-702: All axes? = TRUE, Axis identifier? = FALSE

C-843: All axes? = TRUE, Axis identifier? = FALSE

C-843.PM: All axes? = TRUE, Axis identifier? = FALSE

C-844: All axes? = TRUE, Axis identifier? = FALSE

C-848: All axes? = TRUE, Axis identifier? = FALSE

C-865: All axes? = TRUE, Axis identifier? = FALSE

C-866: All axes? = TRUE, Axis identifier? = FALSE

C-880: All axes? = TRUE, Axis identifier? = FALSE

E-516: If All axes? = TRUE, then Axis identifier? must be TRUE

E-753: If All axes? = TRUE, then Axis identifier? can be FALSE

E-755: If All axes? = TRUE, then Axis identifier? can be FALSE

E-761: If All axes? = TRUE, then Axis identifier? can be FALSE

Mercury™: All axes? = TRUE, Axis identifier? = FALSE

Output	Error out

Remarks — Stops motion of specified axes. To stop a referencing routine (REF, MNL, MPL) or fast scan routine (FSC, FSA etc.), or AutoZero procedure (ATZ), or wave generator run (WGO), use “#24.vi”. STP sets error code 10, call “ERR?.vi” to reset error after STP has been called.

2.2.15. VEL.vi

Valid for — Analog systems, C-702, C-843, C-843.PM, C-844, C-848, C-865, C-866, C-880, C-880K005, E-516, E-710, E-753, E-755, E-761, F-206, M-8X0, Mercury™

Input — System number (1), Without axis ID? (F), No. of digits (4), Axes to set (empty string array), Velocity values (empty num. array, 0), Error in (no error)

Analog: Without axis ID? = FALSE; Velocity unit is µm/sec

C-702: Without axis ID? = FALSE

C-843: Without axis ID? = FALSE

C-843.PM: Without axis ID? = FALSE

C-844: Without axis ID? = FALSE

C-848: Without axis ID? = FALSE

C-865: Without axis ID? = FALSE

C-866: Without axis ID? = FALSE

C-880: Without axis ID? = FALSE, for NanoCube axes command is not valid

C-880K005: VI only supported when called through PI_Multix.vi

E-516: Without axis ID? = FALSE

E-710: Without axis ID? = FALSE. Velocity unit is µm/ms.

E-753: Without axis ID? = FALSE. Velocity unit is µm/s.

E-755: Without axis ID? = FALSE. Velocity unit is µm/s. Command not available for E-755.101.

E-761: Without axis ID? = FALSE. Velocity unit is µm/ms.

F-206: F-206 platform velocity: Without axis ID? = TRUE; velocity of axes A and/or B: Without axis ID? = False; axes K,L,M: command not valid

M-8X0: M-8X0 platform velocity: Without axis ID? = TRUE; velocity of axes A and/or B: Without axis ID? = False

Mercury™: Without axis ID? = FALSE

Output — Error out, Controller error

Remarks — Sets velocity and checks for error. If Without axis ID? is TRUE, then Axes to set is ignored and first field of Velocity values array is used for velocity command. The velocity should not be set to 0. Number of digits is the number of digits after the decimal point in the velocity value(s) that will be sent. Controller error is TRUE if selected system reports error code ≠ 0.

E-516: The VEL command saves the parameters in RAM only. To save the currently valid parameters to flash ROM, where they become the power-on defaults, you must run WPA.vi. Parameter changes not saved with WPA will be lost when the E-516 is powered off.

E-753: Velocity settings made with VEL are present in RAM only and will be reset to default ("Servo Loop Slew Rate" value) when the controller is powered down or rebooted.

E-755: Velocity settings made with VEL are present in RAM only and will be reset to default ("Servo Loop Slew Rate" value) when the controller is powered down or rebooted.

E-761: The VEL command saves the parameters in RAM only. To save the currently valid parameters to flash ROM, where they become the power-on defaults, you must run WPA.vi with "Affected axes" as an empty array. Parameter changes not saved with WPA will be lost when the PC is powered off or the E-761 is rebootet.

2.2.16. VEL?.vi

Valid for	Analog systems, C-702, C-843, C-843.PM, C-844, C-848, C-865, C-866, C-880, C-880K005, E-516, E-710, E-753, E-755, E-761, F-206, M-8X0, Mercury™
Input	System number (1), Axes to query (empty string array), All axes? (F), Axis identifier? (T), Error in (no error) Analog: If All axes? = TRUE, then Axis identifier? can be FALSE; Velocity unit is µm/sec C-702: If All axes? = TRUE, then Axis identifier? can be FALSE C-843: If All axes? = TRUE, then Axis identifier? must be TRUE C-843.PM: If All axes? = TRUE, then Axis identifier? must be TRUE C-844: If All axes? = TRUE, then Axis identifier? must be TRUE C-848: If All axes? = TRUE, then Axis identifier? can be FALSE C-865: If All axes? = TRUE, then Axis identifier? must be TRUE C-866: If All axes? = TRUE, then Axis identifier? must be TRUE C-880: If All axes? = TRUE, then Axis identifier? can be FALSE C-880K005: VI only supported when called through PI_Multix.vi E-516: If All axes? = TRUE, then Axis identifier? must be TRUE E-710: If All axes? = TRUE, then Axis identifier? must be TRUE. Velocity unit is µm/ms. E-753: If All axes? = TRUE, then Axis identifier? can be FALSE. Velocity unit is µm/s. E-755: If All axes? = TRUE, then Axis identifier? can be FALSE. Velocity unit is µm/s. Command not available for E-755.101. E-761: If All axes? = TRUE, then Axis identifier? can be FALSE. Velocity unit is µm/ms. F-206: Velocity of F-206: All axes? = TRUE AND Axis identifier? = FALSE; velocity of axes A,B: All axes? must be FALSE; axes K,L,M: command not valid M-8X0: Velocity of M-8X0: All axes? = TRUE AND Axis identifier? = FALSE; velocity of axes A,B: All axes? must be FALSE Mercury™: If All axes? = TRUE, then Axis identifier? can be FALSE.
Output	Velocity, Error out C-880: NanoCube axes will report velocity = 0 F-206: F-206 velocity: only first field of velocity array is valid M-8X0: M-8X0 velocity: only first field of velocity array is valid
Remarks	Returns velocity setting for specified axes.

2.3. PZT specific VIs ("PZT voltage.llb")

2.3.1. VCO.vi

Valid for	Analog systems, E-516, E-761
Input	System number (1), Axes to command (empty string array), Vel. control mode (empty bool. array, F), Error in (no error)
Output	Error out
Remarks	Sets velocity-control mode for specified axes. Analog: Velocity control mode can only be set for all axes equally. Therefore only first field of Vel. control mode is valid. E-516: The VCO command saves the parameters in RAM only. To save the currently valid parameters to flash ROM, where they become the power-on defaults, you must run WPA.vi. Parameter changes not saved with WPA will be lost when the E-516 is powered off. E-761: The VCO command saves the parameters in RAM only. To save the currently valid parameters to flash ROM, where they become the power-on defaults, you must run WPA.vi with "Affected axes" as an empty array. Parameter changes not saved with WPA will be lost when the PC is powered off or the E-761 is rebootet.

2.3.2. VCO?.vi

Valid for	Analog systems, E-516, E-761
Input	System number (1), Axes to query (empty string array), All axes? (F), Axis identifier? (T), Error in (no error) Analog: All axes? = TRUE; Axis identifier? = FALSE E-516: If All axes? = TRUE, then Axis identifier? must be TRUE E-761: If All axes? = TRUE, then Axis identifier? can be FALSE
Output	Vel. control mode? (T/F), Error out Analog: Only first field of Vel. control mode? is valid.
Remarks	Returns velocity-control mode status for queried axes.

2.3.3. VOL.vi

Valid for	Analog systems, E-761
Input	System number (1), PZT channel (empty string array), PZT voltage (empty num. array, 0), No. of digits (4), Error in (no error) Analog: VI sets control voltage. E-761: PZT channel can be 1 to 4.
Output	Error out
Remarks	Sets absolute PZT voltage for specified PZT channel. No. of digits is the number of digits after the decimal point in the voltage value(s) that will be sent. If the commanded voltage exceeds the voltage limits of the piezo channel, then the command is not executed.

2.3.4. VOL?.vi

Valid for	Analog systems, E-516, E-710, E-753, E-755, E-761, E-816
Input	System number (1), Axes to query (empty string array), All axes? (F), Axis identifier? (T), Error in (no error)
	Analog: If All axes? = TRUE, then Axis identifier? can be FALSE. VI reads control voltage.
	E-516: If All axes? = TRUE, then Axis identifier? must be TRUE
	E-710: If All axes? = TRUE, then Axis identifier? must be TRUE. Axes to query are piezo channel numbers.
	E-753: If All axes? = TRUE, then Axis identifier? can be FALSE. Axes to query are piezo channel numbers.
	E-755: If All axes? = TRUE, then Axis identifier? can be FALSE. Axes to query are piezo channel numbers.
	E-761: If All axes? = TRUE, then Axis identifier? can be FALSE. Axes to query are piezo channel numbers, which can be 1 to 4.
	E-816: All axes? = FALSE, only one axis per command allowed.
Output	Current PZT voltage, Error out
Remarks	Returns current PZT voltage for queried axes.

2.4. Special commands ("Special command.llb")

2.4.1. #24.vi

Valid for	Analog systems, C-702, C-843, C-843.PM, C-844, C-848, C-865, C-866, C-880, C-880K005, E-516, E-753, E-755, E-761, F-206, M-8X0, Mercury™ (but must be present for E-710 also)
Input	System number (1), Error in (no error)
	Analog systems: #24 does not set any error code.
	C-880K005: VI only supported when called through PI_Multix.vi
Output	Error out
Remarks	Stops all motion (by sending the single ASCII character 24). #24 sets error code 10, call "ERR?.vi" to reset error after #24 has been called.

2.4.2. #5.vi

Valid for	Analog systems, C-702, C-843, C-843.PM, C-844, C-848, C-865, C-866, C-880, E-753, E-755, E-761, Mercury™ (but must be present for all other systems also)
Input	System number (1), Error in (no error)
Output	Axis moving? (T/F), Error out
Remarks	Polls the motion status of the connected axes by sending the single ASCII character 5. Connected axes are read from Global2.vi and displayed on the front panel for assignment. Required by "General wait for movement to stop.vi" and "Wait for axes to stop.vi".

Analog: Motion status can only be determined for all connected axes, not for single axes.

F-206: Different coding in answer, please use #5_old.vi

M-8X0: Different coding in answer, please use #5_old.vi

2.4.3. DRR?.vi

Valid for	Analog systems, C-702, C-866, E-710, E-753, E-755, E-761
Input	System number (1), Rec. table IDs (Empty num. array, 0), xo (0), N (100), Nmax (1024), Without parameter? (FALSE), Error in (no error)

Analog: Rec. table IDs, xo, N and Nmax are not valid. Without parameter? must be TRUE.

C-702: Nmax = 262144.

C-866: Nmax = 32,256. If N = -1 all points of the last record are returned.

E-710: Nmax = 32256.

E-753: Nmax = 65,536.

E-755: Nmax = 4096.

E-761: Nmax = 8192.

Output	Data, Names, Sample time, Error out
Remarks	Returns N recorded data points. N must be less than or equal to Nmax. For large N values, communication timeout must be set long enough, otherwise a communication error may occur.

E-761: Recording takes place for all recorder tables as long as the wave generator is running for an arbitrary axis. The assignment of axis and data sources to the recorder tables is as follows:

table 1: axis 1 actual position

table 2: axis 2 actual position

table 3: axis 3 actual position

table 4: analog input voltage (same value as read with TAV?, i.e. contains gain and offset for the analog input, see E-761 User Manual)

E-753: The 65,536 points are in equal shares assigned to the available data recorder tables. By default, the number of tables is 8. It can be reduced by setting the appropriate parameter value, see E-753 User Manual for details.

2.4.4. DRR? and display data.vi

Valid for	Analog systems, C-702, C-866, E-710, E-753, E-755, E-761
Input	System number (1), Rec. table IDs (Empty num. array, 0), xo (0), N (100), Nmax (1024), Without parameter? (FALSE), Error in (no error)

Analog: Rec. table IDs, xo, N and Nmax are not valid. Without parameter? must be TRUE.

C-702: Nmax = 262144

C-866: Nmax = 32,256. If N = -1 all points of the last record are returned.

E-710: Nmax = 32256.

E-753: Nmax = 65,536.

E-755: Nmax = 4096

E-761: Nmax = 8192

Output	Data, Names, Sample time, Error out
Remarks	Returns N recorded data points and displays them in a 2D graph by calling "Show_Save_Load_XY_Data.vi. N must be less than or equal to Nmax. For large N values, communication timeout must be set long enough, otherwise a communication error may occur.

E-761: Recording takes place for all recorder tables as long as the wave generator is running for an arbitrary axis. The assignment of axis and data sources to the recorder tables is as follows:

table 1: axis 1 actual position

table 2: axis 2 actual position

table 3: axis 3 actual position

table 4: analog input voltage (same value as read with TAV?, i.e. contains gain and offset for the analog input, see E-761 User Manual)

E-753: The 65,536 points are in equal shares assigned to the available data recorder tables. By default, the number of tables is 8. It can be reduced by setting the appropriate parameter value, see E-753 User Manual for details.

2.4.5. STA?.vi

Valid for	C-702, C-848, C-880, C-880K005 (but must be present in Special command.llb for all other systems also)
Input	System number (1), Axes to query (empty string array), All axes? (F), Axis identifier? (T), Error in (no error)

C-702: If All axes? = TRUE, then Axis identifier? can be FALSE

C-848: If All axes? = TRUE, then Axis identifier? can be FALSE

C-880: If All axes? = TRUE, then Axis identifier? can be FALSE

C-880K005: VI only supported when called through PI_Multix.vi

Output: Axis status, Error out

C-702: See GCS DLL manual or User manual for supported status bits.

C-848, C-880:

The status word for each axis is a 16-bit register containing the following information (bit encoding is 0 = LSB, 15 = MSB):

Bit #	Description
0	Motion complete flag. This bit is set (1) when the axis trajectory has completed. This flag is only valid for the S-curve, trapezoidal, and velocity contouring profile modes.
1	Wrap-around condition flag. This bit is set (1) when the axis has reached one end of its travel range and has wrapped to the other end of the travel range. Specifically, when traveling in a positive direction past the position +1,073,741,823, the axis will wrap to position -1,073,741,824, and vice-versa. The bit can be reset with the CLR command.
2	Breakpoint reached flag. This bit is set (1) when one of the breakpoint conditions has occurred.
3	Index pulse received flag. This bit is set (1) when an index pulse has been received.
4	Motion error flag. This bit is set (1) when the maximum position error is exceeded. This bit can only be reset when the axis is no longer in a motion

	error condition
5	Positive limit switch flag. This bit is set (1) when the positive limit switch goes active.
6	Negative limit switch flag. This bit is set (1) when the negative limit switch goes active.
7	Command error flag. This bit is set (1) when an erroneous command has been received by the motion control chip.
8*	Servo-control on/off status (1 indicates on, 0 indicates off).
9*	Axis on/off status (1 indicates on, 0 indicates off). The C-848 always has the axis ON.
10*	In-motion flag. This bit is continuously updated and indicates whether or not the axis is in motion: 1 indicates axis is in motion, 0 not in motion.
11*	Reserved (may contain 0 or 1)
12*,13*	Current axis # (13 bit = high bit, 12 bit = low bit). Axis encoding is as follows:

Bit 13	Bit12	MC Axis	C-848 Axis
0	0	1	A
0	1	2	B
1	0	3	C
1	1	4	D

14,15	Reserved (may contain 0 or 1)

C-880K005:

The status word for each axis is a 16-bit register containing the following information (bit encoding is 0 = LSB, 15 = MSB):

Bit #	Description
0	Motion complete flag. Set to 1 when motion is completed. SetMotionCompleteMode determines if this bit is based on the trajectory generator position or the encoder position.
1	Wrap-around condition flag. This bit is set (1) when the actual (encoder) position wraps from maximum allowed position to minimum or vice versa.
2	Breakpoint 1 reached flag. This bit is set (1) when breakpoint 1 is triggered.
3	Capture received flag. This bit is set (1) when a position caputre occures.
4	Motion error flag. This bit is set (1) when a motion error occurs
5	Positive limit switch flag. This bit is set (1) when the positive limit switch goes active.
6	Negative limit switch flag. This bit is set (1) when the negative limit switch goes active.
7	Instruction error flag. This bit is set (1) when an instruction error occurs.
8-10	Reserved, may be 0 or 1.
11	Commutation error flag. This bit is set (1) when a commutation error occurs.
12-13	Reserved, may be 0 or 1.
14	Breakpoint 2 reached flag. This bit is set (1) when breakpoint 2 is triggered.
15	Reserved, may be 0 or 1.

Remarks	Returns axis status (integer). Required by "General wait for movement to stop.vi" and "Wait for axes to stop.vi".

2.4.6. STE.vi

Valid for	Analog systems, C-843, C-843.PM, C-848, C-865, C-866, C-880, E-710, E-753, E-755, E-761
Input	System number (1), Axis to command (empty string), Step size (0), Delay (0), No. of digits (4), Error in (no error) All systems: Delay = 0.
Output	Error out
Remarks	Performs a step-move from, and back to, the current position with specified step size (amplitude). If supported, Delay sets the number of servo loops between position recording (GCS II: Delay must be 0).. No. of digits is the number of digits after the decimal point in the step size (amplitude) values that will be sent. Controller saves a definite number of position values which can be read out with STE?.vi (GCS I) or DRR?.vi (GCS II). Use "General wait for movement to stop.vi" before calling "STE?.vi" or "DRR?.vi" to make sure that motion has finished before reading back the saved values. For an impulse-move, see "IMP.vi".

Analog: Use DRR?.vi or DRR? and display data.vi to read position values back.

C-843: Controller saves up to 32,640 position values for all 4 channels in sum. Use STE?.vi to read position values back.

C-843.PM: Controller saves up to 32,640 position values for all 4 channels in sum. Use STE?.vi to read position values back.

C-848: Controller saves 1024 position values. Use STE?.vi to read position values back.

C-865: Controller saves up to 32,640 position values. Use STE?.vi to read position values back.

C-866: Controller saves up to 32,256 position values. STE will overwrite DRC settings of Rec. table 1 to record actual position values. Use DRC to define additional record options for Rec. table no. 2 to 4. Record table rate is reset to 1 by STE. Use STE?.vi to read position values back or DRR? to read all Rec. tables back. You can also use MVR in combination with DRC to record values of a step motion. Use DRR? to read values back then.

C-880: Controller saves 1024 position values. Use STE?.vi to read position values back.

E-710: Controller saves 8192 position values. "Table Rate" parameter, set with SPA, is used as sampling interval instead of Delay. Caution: Table Rate parameter influences Wave Generator, not only STE. Use STE?.vi to read position values back.

E-753: Controller saves up to 65,536 position values. Use DRR?.vi or DRR? and display data.vi to read recorded values back. The number of servo cycles used for data recording depends on the setting made with RTR.
Motion commands are not allowed when the wave generator is active or the analog input is used for target generation.

E-755: Controller saves 4,096 position values. Use DRR?.vi or DRR? and display data.vi to read recorded values back.

E-761: Controller saves 8192 position values. The number of servo cycles used for data recording depends on the setting made with RTR. Use STE?.vi to read position values back.

2.4.7. TPC?.vi

Valid for	E-710, E-753, E-755, E-761 (but must be present for Analog systems, E-516 and E-816 also)
Input	System number (1), Error in (no error)
Output	Number of piezo channels, Error out
Remarks	Returns the number of available piezo channels.

2.5. Old commands and commands with alternate implementations (“Old commands.llb”)

2.5.1. #5_old.vi

Valid for	F-206, M-8X0 (but must be present for all other systems also)
Input	System number (1), Error in (no error)
Output	Overall system moving? (T/F), Sep. Axis 1 moving? (T/F), Sep. Axis 2 moving? (T/F), Error out
Remarks	Polls the motion status of the F-206/M-8X0 and/or up to 2 additional connected axes by sending the single ASCII character 5. Required by “General wait for movement to stop.vi”.

2.5.2. Wait for hexapod system axes to stop.vi

Valid for	F-206, M-8X0 (but must be present for all other systems also)
Input	System number (1), All axes? (T), Axes to wait for (empty string array), Stop refnum (F), Local stop (F), Error in (no error) To wait for the hexapod to stop, only one hexapod axis (X, Y, Z, U, V or W) needs to be commanded, because the VI cannot distinguish between the different hexapod axes. F-206: <u>Axes to wait for</u> can be any of X, Y, Z, U, V, W, A, B, K, L, M M-8X0: <u>Axes to wait for</u> can be any of X, Y, Z, U, V, W, A, B
Output	Error out
Remarks	This vi waits for the specified axes of a PI hexapod system (hexapod axes X, Y, Z, U, V, W and separate axes A, B) to stop using #5 polling. If a NanoCube axis (K, L or M) is commanded, the VI will return immediately. If one of the hexapod axes (X, Y, Z, U, V or W) is commanded, it will wait for all six hexapod axes to stop. It returns immediately if a communications error occured, or if <u>Local stop</u> or <u>Stop refnum</u> is TRUE. When using as a sub-VI, use <u>Refnum stop</u> to stop VI from caller. Required by “General wait for movement to stop.vi”.

2.6. File handling VIs ("File handling.llb")

2.6.1. Array File.vi

Valid for	Analog systems, C-880, E-761, F-206, M-8X0
Input	Path (empty path), Read (F)/Delete (F), ArrayName (empty string), Error in (no error)
Output	Array names, Error out
Remarks	This vi checks the names of all arrays in a data file or deletes a given array from a data file.

2.6.2. File handler.vi

Valid for	Analog systems, C-702, C-866, C-880, E-710, E-753, E-755, E-761, F-206, M-8X0
Input	Path in (empty path), Read (F) or write (T)? (F), With dialog? (F), Write new file? (F), Default file name (empty string), Extension (txt)
Output	Path out, Cancelled? (T/F), Data added? (T/F)
Remarks	This vi handles file name selections with or without a user interface. Files can be read or written. Path in is the path to the file to read or write. Extension is the file extension for the file to write (e.g. txt, jpg). If Read (F) or write (T) is TRUE, Extension must be given and entry must not have a dot. If With dialog? is TRUE, in every case a dialog box will pop up where the file to read or write can be selected. Default file name is used for naming suggestions if a dialog pops up. If Read (F) or write (T)? is TRUE and Write new file? is TRUE, a dialog box will pop up if the selected file name already exists. If Write new file? is FALSE and the selected file name already exists, a dialog box will pop up to ask if data should be added. Data added? indicates if data was added to an existing file. Cancelled? indicates if the user has cancelled the operation. Path out is NotAPath if operation was cancelled or not successful and contains the selected path for the file which was read or written if the operation was successful.

2.6.3. GetDataFormat.vi

Valid for	Analog systems, C-702, C-866, C-880, E-710, E-753, E-755, E-761, F-206, M-8X0
Input	IOSource (Read (F)/Write (F), Path (empty path), ArrayName (empty string), Datastream (empty string)), Error in (no error)
Output	Header out (Separator, NDim, Remarks), DataOK, Found Header, Data Type, NData, Names out, Sample time, Error out
Remarks	This vi checks the format of a data file. See separate manual "GCSData_User_SM146E.pdf" and control descriptions in the diagram for more information.

2.6.4. MatrixIO.vi

Valid for	Analog systems, C-866, C-880, E-753, E-755, E-761, F-206, M-8X0
Input	IOSource (Read (F)/Write (F), Path (empty path), ArrayName (empty string), Datastream (empty string)), Header in (Separator (\t), NDim (0), Remarks (empty string)), Data names (XName (empty string), YName

	(empty string), ZName (empty string)), XArray in (empty num. array), YArray in (empty num. array), ZMatrix in (empty 2D num. array), Sample time in (0), (Error in (no error)
Output	Datastream out, Header out (Separator, NDim, Remarks), Data names out (XName, YName, ZName), XArray out, YArray out, ZMatrix out, Sample time out, Error out
Remarks	This vi reads or writes data files in matrix format. See separate manual “GCSData_User_SM146E.pdf” and control descriptions in the diagram for more information.

2.6.5. TableIO.vi

Valid for	Analog systems, C-702, C-866, C-880, E-710, E-753, E-755, E-761, F-206, M-8X0
Input	IOSource (Read (F)/Write (F), Path (empty path), ArrayName (empty string), Datastream (empty string)), Header in (Separator (\t), NDim (0), Remarks (empty string)), Names in (empty string array), Table in (empty 2D num. array), Sample time in (0), (Error in (no error)
Output	Datastream out, Header out (Separator, NDim, Remarks), Names out, Table out, Sample time out, Error out
Remarks	This vi reads or writes data files in table format. See separate manual “GCSData_User_SM146E.pdf” and control descriptions in the diagram for more information. Sub-VI for “DRR?.vi”.

2.7. Limit- and reference-specific commands (“Limits.llb”)

2.7.1. TMN?.vi

Valid for	Analog systems, C-702, C-843, C-843.PM, C-844, C-848, C-865, C-866, C-880, E-710, E-753, E-755, E-761, Mercury™
Input	System number (1), Axes to query (empty string array), All axes? (F), Axis identifier? (T), Error in (no error)

Analog: If All axes? = TRUE, then Axis identifier? can be FALSE

C-702: If All axes? = TRUE, then Axis identifier? can be FALSE

C-843: If All axes? = TRUE, then Axis identifier? must be TRUE

C-843.PM: If All axes? = TRUE, then Axis identifier? must be TRUE

C-844: If All axes? = TRUE, then Axis identifier? must be TRUE

C-848: If All axes? = TRUE, then Axis identifier? can be FALSE

C-865: If All axes? = TRUE, then Axis identifier? must be TRUE

C-866: If All axes? = TRUE, then Axis identifier? must be TRUE

C-880: If All axes? = TRUE, then Axis identifier? can be FALSE

E-710: If All axes? = TRUE, then Axis identifier? must be TRUE

E-753: If All axes? = TRUE, then Axis identifier? can be FALSE.

E-755: If All axes? = TRUE, then Axis identifier? can be FALSE. Command not available for E-755.101.

E-761: If All axes? = TRUE, then Axis identifier? can be FALSE

Mercury™: If All axes? = TRUE, then Axis identifier? can be FALSE

Output Minimum travel limit, Error out

Remarks Returns minimum (low-end) travel limit (if present, position of negative limit switch, or value of negative soft limit, if set, whichever is higher).

E-761: Get the minimum accessible position value, i.e. the value of the "Range min limit" parameter (ID 0x07000000). Note: The minimum position which can be commanded depends either on the "Range min limit" parameter or—if it is greater than the "Range min limit" parameter value— on the value of the negative soft limit set with NLM.

2.7.2. TMX?.vi

Valid for Analog systems, C-702, C-843, C-843.PM, C-844, C-848, C-865, C-866, C-880, E-710, E-753, E-755, E-761, Mercury™

Input System number (1), Axes to query (empty string array), All axes? (F), Axis identifier? (T), Error in (no error)

C-702: If All axes? = TRUE, then Axis identifier? can be FALSE

C-702: If All axes? = TRUE, then Axis identifier? can be FALSE

C-843: If All axes? = TRUE, then Axis identifier? must be TRUE

C-843.PM: If All axes? = TRUE, then Axis identifier? must be TRUE

C-844: If All axes? = TRUE, then Axis identifier? must be TRUE

C-848: If All axes? = TRUE, then Axis identifier? can be FALSE

C-865: If All axes? = TRUE, then Axis identifier? must be TRUE

C-866: If All axes? = TRUE, then Axis identifier? must be TRUE

C-880: If All axes? = TRUE, then Axis identifier? can be FALSE

E-710: If All axes? = TRUE, then Axis identifier? must be TRUE

E-753: If All axes? = TRUE, then Axis identifier? can be FALSE.

E-755: If All axes? = TRUE, then Axis identifier? can be FALSE. Command not available for E-755.101.

E-761: If All axes? = TRUE, then Axis identifier? can be FALSE

Mercury™: If All axes? = TRUE, then Axis identifier? can be FALSE

Output Maximum travel limit, Error out

Remarks Returns maximum (high-end) travel limit (if present, position of positive limit switch or value of positive soft limit, if set, whichever is lower).

E-761: Get the maximum accessible position value, i.e. the value of the "Range max limit" parameter (ID 0x07000001). Note: The maximum position which can be commanded depends either on the "Range max limit" parameter or—if it is smaller than the "Range max limit" parameter value— on the value of the positive soft limit set with PLM.

2.8. Commands for Optical or Analog Signals (“Optical or Analog Input.llb”)

2.8.1. TSC?.vi

Valid for	E-710, E-753, E-755, E-761 (but must be present for Analog systems, E-516 and E-816 also)
Input	System number (1), Error in (no error)
Output	Number of sensor channels, Error out
Remarks	Returns the number of available sensor channels. E-753: The response comprises all ADC channels of the device: the "genuine" sensor (capacitive sensor integrated in the mechanics) and the "general purpose" analog input.

2.9. Support VIs for scanning algorithms (“Scan support.llb”)

2.9.1. Axis names.vi

Valid for	Analog systems, C702, C-866, C-880, E-710, E-753, E-755, E-761, F-206, M-8X0
Input	Names (empty string array)
Output	X axis name, Y axis name, Z axis name
Remarks	Checks if "Names" contains three strings for axis names. If this is not the case, it assigns "X Values", "Y Values" and/or "Z Values" as the missing axis name.

2.10. Analog controller VIs (“Analog control.llb”)

2.10.1. Analog FGlobal.vi

Valid for	Analog systems (but must be present for all other systems, also)
Input	System no. (1), Read(F)/Write (TRUE), VI ref in
Output	VI ref out
Remarks	This VI works as a functional global variable for VI references

2.10.2. Analog functions (dyn).vi

Valid for	Analog systems
Input	System number (1), Command (INITIALIZE), String (empty string), AI Task, AO Task, W_Wave settings, Continuously? (TRUE), Error in (no error)
Output	String output, Boolean output, Error out
Remarks	Executes DAQmx functions and other tasks, depending on String to send. VI is called dynamically from “Analog Functions.vi”.

2.10.3. Analog functions.vi

Valid for	Analog systems (but must be present as a Dummy VI for all other systems also)
Input	System number (1), String to send (empty string), type specifier VI Refnum, AI Task, AO Task, Waveform to write, Continuously? (TRUE), Error in (no error)
Output	Command, String output, Boolean output, Error out
Remarks	Calls Analog Functions (dyn).vi functions dynamically during runtime, depending on String to send.

2.10.4. Analog functions.vi

Valid for	Analog systems (but must be present for all other systems also) --- Dummy VI
Input	System number (1), String to send (empty string), type specifier VI Refnum, AI Task, AO Task, Waveform to write, Continuously? (TRUE), Error in (no error)
Output	Command, String output, Boolean output, Error out
Remarks	Dummy VI

2.10.5. Analog Receive String.vi

Valid for	Analog systems (but must be present for all other systems also)
Input	System number (1), Read/Write (T) (FALSE), Ini (False), Error in (no error)
Output	String out, Strings out, Error out
Remarks	Works as an old style global variable for String out.

2.10.6. Available Analog Commands.ctl

Valid for	Analog systems (but must be present for all other systems also)
Input	None
Output	None
Remarks	Type definition for available analog commands.

2.10.7. Build analog reply string.vi

Valid for	Analog systems
Input	System number (1), All axes? (F), With values? (T), Values only (F), Affected axes (empty string array), Values (empty num. array), Read/Write (T) (FALSE), Ini (False)
Output	String
Remarks	Builds a query-command reply string by combining axes and values into a string following GCS II conventions. If With values? is FALSE, only axes are returned. For Values only? = TRUE, only values are returned without axes.

2.10.8. Build analog value array.vi

Valid for	Analog systems
Input	System number (1), Initialize? F), Waveform? (F), Absolute (T)/Relative (TRUE), Voltage? (T), W_Wave? (F), Range check? (T), Values in (empty num. array), Waveform to write, Error in (no error)
Output	Axes out, Values out, Waveform out, T, Error out
Remarks	Returns an array of numbers or waveforms and axes parsed from String, which must follow the GCS II conventions. If Initialize? is TRUE, the VI is initialized with Values in. If Absolute (T)/Relative is TRUE, the values in String are considered to be absolute values, for FALSE they are considered relative to the current voltage/position. If Voltage? is FALSE, the conversion factor V/µm is read from “GlobalAnalog.vi”, the values in String are considered to be position values and Values out/Waveform out contains voltage information. If Voltage? is TRUE, the values in String are considered to be voltage values. T is the total time for Waveform out.

2.10.9. Build analog waveform.vi

Valid for	Analog systems
Input	System number (1), Index of axis (0), Start voltage (0), Target voltage (0), Error in (no error)
Output	Waveform, T(s), N, HyperBit Bits, Start=Target?, Error out
Remarks	Returns a waveform calculated by start and target voltage for the given axis.

2.10.10. Get Terminal Name with Device Prefix.vi (NI Example VI)

Valid for	Analog systems
Input	Task/channels in, Local Terminal Name (empty string), Error in (no error)
Output	Task out, Terminal Name with Device Prefix, Error out
Remarks	Sample VI from NI which takes in a DAQmx Task and a terminal and converts this to a terminal which includes the device prefix specified in the DAQmx Physical Channel Control. For example: DAQmx Task/channels in=Dev1/ai1:2 Terminal=ai/StartTrigger Device+Terminal=/Dev1/ai/StartTrigger

2.10.11. Global Analog.vi

Valid for	Analog systems (but must be present for all other systems also)
Input	None
Output	None
Remarks	A global variable which contains setup information for analog systems.

2.10.12. HyperBit Core.vi

Valid for	Analog systems
Input	PWM rate (Hz) (20000), Bit rate (Hz) (1000), Physical bit resolution (12), AO Vmin (0), AO Vmax (10), Password (empty string), Waveform in, Error

	in (no error)
Output	Voltage resolution, Incoming waveform frequency, Hyperbitted waveform Out, Error out
Remarks	PI HyperBit Core VI. U.S. Patent 6,950,050; international patents pending. Calculates hyperbitted waveform for incoming waveform with given settings.

2.10.13. HyperBit Frame.vi

Valid for	Analog systems
Input	System no. (1), PWM rate (Hz) (20000), Bit rate (Hz) (1000), Waveform in, Error in (no error)
Output	Waveform Out, Error out
Remarks	PI HyperBit Frame VI. U.S. Patent 6,950,050; international patents pending. Calls HyperBit Core.vi with given settings and returns hyperbitted waveform.

2.10.14. HyperBit settings.vi

Valid for Analog systems

Input System no. (1), Read/Write(T) (False),HyperBit settings in, Error in (no error)

The Bit rate should be at least double the resonant frequency of the stages connected. If you have different types of stages connected, select the Bit rate to be at least double the maximum resonant frequencies of all stages.

The PWM rate must be at least double the Bit rate. If you experience performance problems due to too much bus traffic, decrease the PWM rate value.

HB StayOnPos: Select if the position reached with HyperBit ON is to hold. If this control is TRUE and HyperBit is switched ON, LabVIEW will continue to output a hyperbitted target position after the axes have reached their target position. In that way position values which do not correspond to the physical bit resolution of the DAC card can be hold. If this control is FALSE, the analog output will stop after having reached the target position and if the target position does not correspond to the physical bit resolution of the DAC board, the resulting final position will be not as exact. If HyperBit is switched OFF, this control does not have any effect.

Output HyperBit settings out, Error out

Remarks Sets or returns HyperBit settings, depeding on Read/Write (T).

If you generate too many HyperBits by setting the PWM rate too high, the rate capacity of the card or the load capacity of the computer's bus can be exceeded. This may result in a runtime error, or sluggish operation during HyperBit analog output. Solutions: set your VIs to request a lower PWM rate; check your NIMax configuration settings to ensure DMA is being used; or use a faster card. Keep in mind that DAC granularity is only one of many possible limiters to system resolution. Ambient noise and vibration and electronic/amplifier noise, for example, also limit what your system can achieve. There is no point generating HyperBits below the fundamental noise floor.

2.10.15. Increase waveform by n values.vi

Valid for	Analog systems
Input	Waveform in, Percent (10)

Output	Waveform out
Remarks	Increases a given waveform by N percent of the original length. The additional points are equal to the last point of the incoming waveform.

2.10.16. Initialize Global Analog.vi

Valid for	Analog systems
Input	System no. (1), Error in (no error)
Output	Error out
Remarks	Initializes Global Analog according to the given system no.

2.10.17. Multiboard Trigger handling.vi

Valid for	Analog systems
Input	AI Task in, AO Task in, Error in (no error)
Output	AI Task out, AO Task out, Equal?, Error out
Remarks	This vi checks if AI and AO use the same DAC device and if this is the case, it uses the DAQmx Start Trigger (Digital Edge).vi for triggering. If Equal? returns FALSE, please implement your own triggering functions here.

2.10.18. PI Square Wave.vi

Valid for	Analog systems
Input	Samples (128), amplitude (1), f (7.8125E-3), phase in (0), reset phase (T), duty cycle (%) (50)
Output	Square wave, phase out, error
Remarks	Generates an array specifying a square wave.

2.10.19. Resulting resolution.vi

Valid for	Analog systems
Input	Supported AO voltage ranges (empty num. array), Lower voltage limit (V) (0), Upper voltage limit (V) (10), AO.Resolution (0), error in (no error)
Output	Vmin, Vmax, Gain, Res. AO.Resolution, Vmin G0, Vmax G0, Res. AO. Resolution G0, error out
Remarks	Calculates which analog output voltage range is best for the given Lower and Upper voltage limit and returns the appropriate gain and the resulting resolution value. It also calculates range and resolution if setting a gain value for the analog output channel is not supported by the DAC card (i.e. gain must be zero).

2.10.20. Substract analog offset.vi

Valid for	Analog systems
Input	System no. (1), Single axis? (F), Values in (empty num. array), Index of axis (empty num. array), Error in (no error)
Output	Values out, First value out, Error out
Remarks	Calculates an offset-corrected array of numerical values. If Single axis? is TRUE, Values in are considered to belong to one axis only and the

correction factor of that axis is used. If Single axis? is FALSE, each value in Values in is considered to belong to another axis and must contain as many values as there are axes connected. VI also checks whether Values in are in allowed voltage range and if they are, it coerces Values out to be so also.

2.10.21. Write analog waveform.vi

Valid for	Analog systems
Input	System no. (1), Continuously? (T), Axes to move (empty string array), Waveform to write, Error in (no error)
Output	Error out
Remarks	Outputs the given waveform for Axes to move once or continuously.

2.11. Support VIs ("Support.llb")

Support VIs are sub-VIs for command VIs which make certain programming tasks more convenient. They can also be used for building main programs.

Caution: Please do not change these VIs, as that might cause the command VIs that use them to fail.

2.11.1. Assign booleans from string to axes.vi

Valid for	All Systems
Input	System number (1), Queried axes (empty string array), All axes queried? (F), Input string (empty string), Error in (no error)
Output	Booleans(T/F), Error out
Remarks	This VI assigns numerical values from input string to boolean values for queried axes. If All axes? is TRUE, connected axes are read from Global2.vi and displayed on the front panel for assignment. Example: An input string like "A=0SpaceLinefeedB=1Linefeed" or "0SpaceLinefeed1Linefeed" will be converted to an output array consisting of two values "FALSE; TRUE".

2.11.2. Assign NaN for chosen axes.vi

Valid for	Analog systems, C-702, C-843, C-843.PM, C-848, C-865, C-866, C-880, E-753, E-755, F-206, M-8X0, Mercury™
Input	Queried axes (empty string array), Values (empty num. array), Axes subset (empty string array), Value to set (NaN)
Output	New values
Remarks	This VI returns "NaN" or any given Value to set for the given axes subset.

2.11.3. Assign values from string to axes.vi

Valid for	All systems
Input	System number (1), Queried axes (empty string array), All axes queried? (F), Axes related? (T), Input string (empty string), Error in (no error)
Output	Values, Strings, Error out
Remarks	This VI assigns numerical values and/or single lines from input string to queried axes. If All axes? is TRUE, connected axes are read from Global2.vi and displayed on the front panel for assignment. If All axes? is TRUE and Axes related? is FALSE, item names from Input string are displayed instead of connected axes.

2.11.4. Boolean array calculations.vi

Valid for	All systems
Input	Array1 (empty bool. array), Array2 (empty bool. array), Array3 (empty bool. array), Operator (AND)
Output	Array out
Remarks	This vi performs a boolean operation of up to three boolean input arrays. The difference to LabVIEWs own boolean operators is that the input arrays can have different sizes. The missing elements are considered to be

FALSE elements and the resulting array contains the maximum number of elements.

2.11.5. Build channel query command substring.vi

Valid for	Analog systems, E-516, E-710, E-753, E755, E-761, E-816
Input	System number (1), Channels to query in (empty string array), Query all channels? (F), With space? (F), Channel identifer? (T), Channel type (piezo), Error in (no error)
Output	Command substring, Channels to query out, Number of rows, Error out
Remarks	This VI builds a query command substring for channel query commands. If All channels? is TRUE, channels to command are determined in a controller specific way and returned in Channels to query out, otherwise Channels to query out is identical with Channels to query in. Number of rows is size of the Channels to query out array. If Channel identifier? is FALSE, command substring is an empty string (e.c. for systems which accept commands like VMA? without channel IDs). If With space? is TRUE, a space character is added between the channel identifiers.

2.11.6. Build command substring.vi

Valid for	All systems
Input	Affected axes (empty string array), No. of digits (4), Parameters (empty num. array, 0), Parameters (hex.) (empty hex. array), Parameter no. format (Decimal: FALSE) (F), With space? (F)
Output	Command substring
Remarks	This VI builds a command substring by combining axis identifier and parameter. If parameter number is in decimal format, use Parameters input, for hexadecimal parameter numbers use Parameters (hex.) input and switch Parameter no. format to TRUE. Do not mix decimal and hex. parameter numbers in one call. No. of digits is the number of digits after the decimal point in the parameter value(s) that will be sent. Example: For Affected axes = A; B, Parameters = 1.2342; 2.3 and No. of digits = 3 the resulting string is “SpaceA1.234SpaceB2.300”.

2.11.7. Build num command substring.vi

Valid for	All systems
Input	No. of digits (4), Num 1 (empty num. array, 0), Num 2 (empty num. array, 0)
Output	Command substring
Remarks	This VI builds a command substring by combining Num1, Space and Num2. No. of digits is the number of digits after the decimal point in the Num 1/2 value(s) that will be sent. Example: For Num 1 = 1.24; 3.25456, Num 2 = 5.0; 7.4321 and No. of digits = 3 the resulting string is “Space1.240Space5.000Space3.255Space7.432"

2.11.8. Build query command substring.vi

Valid for	All systems
Input	System number (1), Axes to query in (empty string array), Query all axes?

	(F), With space? (F), Axis identifer? (T),
Output	Command substring, Axes to query out, Number of rows
Remarks	This VI builds a query command substring. If All axes? is TRUE, connected axes are read from “Global2.vi” and returned in Axes to query out, otherwise Axes to query out is identical with Axes to query in. Number of rows is size of the Axes to query out array. If Axis identifier? is FALSE, command substring is an empty string (e.c. for systems which accept commands like POS? without axis IDs). If With space? is TRUE or system supports GCS II, a space character is added between the axes identifiers. Example: If axes A;B;C;D are connected to the system to command, Axes to query in is A;B;D, Query all axes? is TRUE and Use Axis identifier? is TRUE, resulting Command substring is “ABCD”, Number of rows is 4 and Axes to query out is A;B;C;D. If With space? is TRUE, the resulting Command substring is “A B C D”.

2.11.9. Build stringplusnum substring.vi

Valid for	All systems
Input	Sequence (String1String2String3Value1Value2), String1 (empty string array), String2 (empty string array), String3 (empty string array), Value1 (empty num. array, 0), Value2 (empty num. array, 0), No. of digits Value1 (6), No. of digits Value2 (6), Input selection (T,T,T,T,F), Error in (no error)
Output	Substring, Error out
Remarks	This vi builds a command substring by combining up to three strings and two values in the given order.

2.11.10. Commanded axes connected?.vi

Valid for	All systems
Input	System number (1), Commanded axes (empty string array), Error in (no error)
Output	Controller error (T/F), Error out
Remarks	This VI checks if Commanded axes are a subset of all connected axes (read from “Global2 (Array).vi”) and returns Controller error TRUE if this is not the case. Connected axes are defined by “Define connected axes.vi”, which is called by “XXX_Configuration_Setup.vi” automatically. White space strings in Commanded axes are ignored.

2.11.11. Convert num array to string.vi

Valid for	All systems
Input	Number of digits (4), Num. values (empty num. array)
Output	Output string
Remarks	This vi converts an array of numerical values to a space separated output string. The difference to LabVIEW's native Array to Spreadsheet String function is that no carriage return or newline is added.

2.11.12. Convert num value to syntax selection.vi

Valid for	All systems
Input	GCS syntax version (1,00)

Output	Syntax
Remarks	This VI converts a numerical value to the corresponding GCS syntax version.

2.11.13. Count occurrences in string.vi

Valid for	All systems
Input	Input string (empty string), Expression (empty string)
Output	Occurrences
Remarks	This VI counts, how often an expression occurs in a string.

2.11.14. Cut out additional spaces.vi

Valid for	All systems
Input	Mode (All Spaces), String (empty string)
Output	String out
Remarks	Searches for spaces in String and cuts them out, depending on Mode.

2.11.15. Define axes to command from boolean array.vi

Valid for	All systems
Input	Axes to query (empty string array), Command axis? (empty bool. array, F)
Output	Axes to command, Remaining axes
Remarks	This VI returns only those axis IDs from the Axes to query array in the Axes to command array which have a boolean value TRUE in the Command axis? array, and all remaining axes in the Remaining axes array.

2.11.16. GCSTranslateError.vi

Valid for	All systems
Input	Error in (no error)
Output	Error out, GCS Error?, Error description
Remarks	Returns if error in contains a GCS error code and if this is the case, it displays the corresponding error message and appends it to source in error out.

2.11.17. General wait for movement to stop.vi

Valid for	All systems
Input	System no. (1), Axes to wait for (empty string array), All axes? (T), Polling cycle time, ms (1), Additional wait time, ms (0), Error in (no error) F-206: VI will not wait for INI procedure to complete. M-8X0: VI will not wait for INI procedure to complete.
Output	Error out
Remarks	This VI waits for the specified axes to stop. An additional wait time can be specified. The wait method depends on the system to command. “XXX_Configuration_Setup.vi” (with XXX being the product name of your system) must be run before running this vi. Requires “Wait for axes to stop.vi”, “#5.vi”,“STA?.vi”, “#5_old.vi”, “ONT?.vi” and “Wait for hexapod

system axes to stop.vi” to be present.

2.11.18. Get all axes.vi

Valid for	All systems
Input	System no. (1)
Output	Conn. Axes
Remarks	This VI reads all connected axes for given system from "Global2 (Array).vi". Connected axes are defined by "Define connected axes.vi", which is called by "XXX_Configuration_Setup.vi" automatically.

2.11.19. Get arrays without blanks.vi

Valid for	All systems
Input	String array in (empty string array), Values in (empty num. array), Booleans in (empty bool. array, F), Array size in (0)
Output	String array out, Values out, Booleans out, Array size out
Remarks	Returns the string array and related values and boolean arrays without white space string fields.

2.11.20. Get lines and values from string.vi

Valid for	All systems
Input	Array size (0), Input string (empty string)
Output	Numerical values, Strings
Remarks	This VI returns numerical values and single lines from input string without any axis assignment. If number of lines/values (Array size) is known, algorithm is faster, otherwise Array size = 0 should be used. Sub-VI for “VST?.vi” and “STE?.vi”.

2.11.21. Get lines from string.vi

Valid for	All systems
Input	Array size (0), Input string (empty string)
Output	Strings
Remarks	This VI returns single lines from input string. If number of lines (Array size) is known, algorithm is faster, otherwise Array size = 0 should be used. Sub-VI for “VST?.vi”.

2.11.22. Get string array size without blanks.vi

Valid for	All systems
Input	String array (empty string array)
Output	Corrected array size
Remarks	This VI returns the size of a string array without counting white space strings.

2.11.23. How often does string contain regular expression.vi

Valid for	All systems
Input	Regular expression (empty string), String (empty string)
Output	Number
Remarks	This VI returns a count of the occurances of a regular expression in a string.

2.11.24. Increase array size.vi

Valid for	All systems
Input	Size (0), Array in (empty num. array, NaN), Only if Array is not empty?
Output	Array out
Remarks	If size of Array in is smaller than Size, this VI increases the size of Array in to Size. If Array in is an empty array and Only if Array is not empty? is FALSE, VI builds an array of zeros with the size of Size.

2.11.25. Return single characters from string.vi

Valid for	All systems
Input	Input string (empty string), Invert order (F), Error in (no error)
Output	Character array (empty string array), Error out
Remarks	Get single characters from input string.

2.11.26. Return space.vi

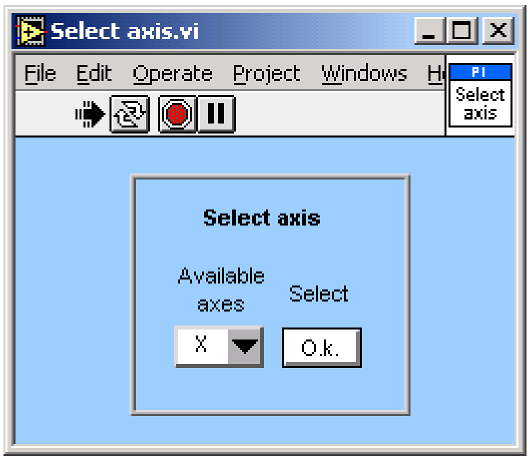
Valid for	All systems
Input	System no. (1), With space? (F)
Output	String out, Space returned?
Remarks	This VI returns a space character in String out if With space? is TRUE or GCS syntax version is higher than 1.0.

2.11.27. Round with options.vi

Valid for	All systems
Input	No. of digits to round to (2), Round mode selection (Round to nearest), Numeric in (0), Num array in (empty num. array)
Output	Numeric out, Num array out
Remarks	Rounds Numeric in and Num array in according to No. of digits to round to and Round mode selection.

2.11.28. Select axis.vi

Valid for	All systems
Input	System number (1)
Output	Selected axis, Index of axis in Global2
Remarks	This VI reads all connected axes from Global2 and writes them into a menu ring control for selection. The selected axis and it’s index in Global2 are returned.



2.11.29. Select values for chosen axes.vi

Valid for	All systems
Input	Queried axes (empty string array), Values (empty num. array), Axes subset (empty string array)
Output	Values subset
Remarks	This VI returns only values for the given axes subset.

2.11.30. Select with boolean array input.vi

Valid for	All systems
Input	Size (0), T string (empty string), F string (empty string), T/F (empty boolean array)
Output	String array out
Remarks	This vi returns a string array of a given size with T string and F string, depending on the boolean value at the corresponding index of T/F.

2.11.31. Selection to string array.vi

Valid for	All systems
Input	Selection array (empty Menu Ring array, 0), String input (empty string array)
Output	String array
Remarks	This vi returns a string array which contains strings according to the selected value of String input. Example: For Selection array = (2,0,1) and String input = (A,B,C) the resulting String array is (C,A,B).

2.11.32. Substract axes array subset from axes array.vi

Valid for	All systems
Input	Axes to query (empty string array), Axes subset (empty string array)
Output	Axes to command, All present?
Remarks	This VI returns only these axes IDs from the Axes to query array which are **not** present in the Axes subset array. If no axes IDs are returned, All present? is TRUE. Needed by “Define axes to command from boolean array.vi”.

2.11.33. Wait for axes to stop.vi

Valid for	C-702, C-843, C-843.PM, C-844, C-848, C-865, C-880, E-753, E-755, Mercury™ (but must be present in Support.llb for all other systems also)
Input	System number (1), Axes to wait for (empty string array), With status bit polling? (F), Polling cycle time, ms (400), Stop refnum (F), Local stop (F), Error in (no error) C-702: With status bit polling? = FALSE C-843: With status bit polling? = FALSE C-843.PM: With status bit polling? = FALSE C-844: With status bit polling? = FALSE C-880: With status bit polling? = TRUE E-753: With status bit polling? = FALSE E-755: With status bit polling? = FALSE Mercury™: With status bit polling? = FALSE
Output	Error out
Remarks	This VI waits for the specified axes to stop using #5 polling. It also stops if a communication error occured, Stop refnum or Local stop is TRUE. Requires "STA?.vi" to be present. Required by "General wait for movement to stop.vi". When using as a sub-VI, use Stop refnum to stop VI from caller.

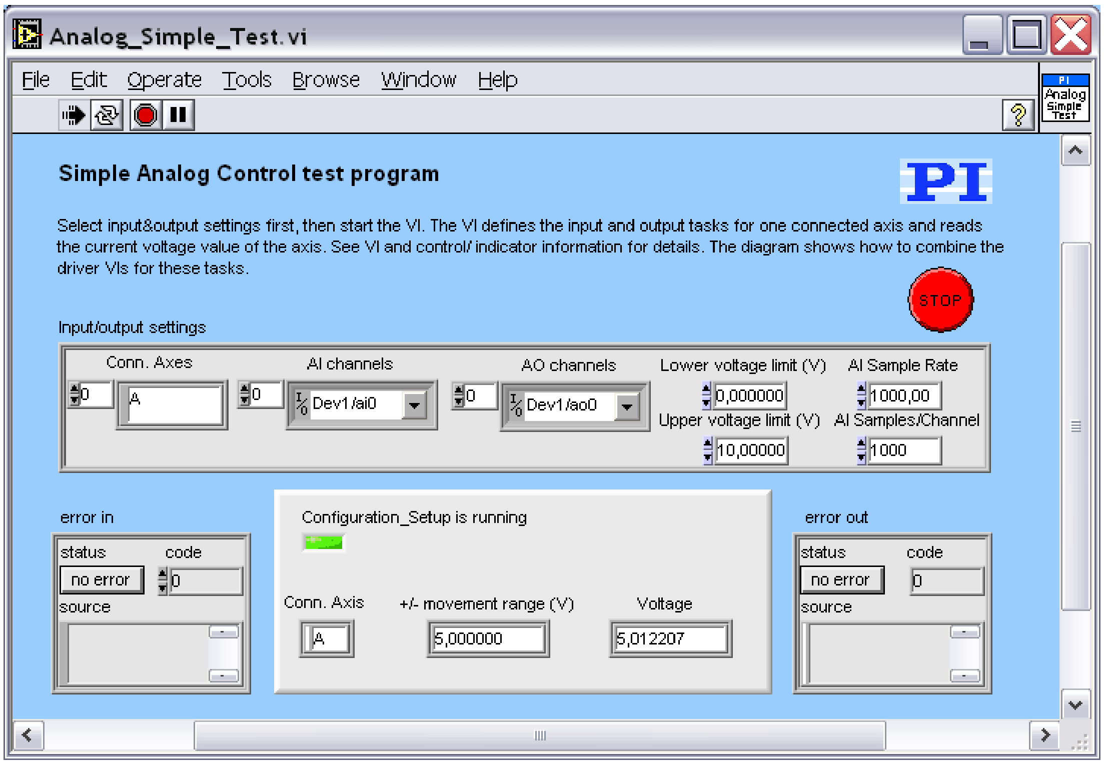
3. High Level VIs

3.1. Analog_Simple_Test.vi

This simple test VI is a stand-alone sample application. Use the *Help→Show Context Help* menu sequence in the LabVIEW environment to display the *Context Help* window with the VI and control/indicator descriptions.

Select Input/output settings first, then start the VI. The VI defines the input and output tasks for one connected axis and reads the current voltage value of the axis (VOL?). See VI and control/ indicator information for details.

The diagram shows how to combine the driver VIs for these tasks.



Simple Analog Control test program

Select input&output settings first, then start the VI. The VI defines the input and output tasks for one connected axis and reads the current voltage value of the axis. See VI and control/ indicator information for details. The diagram shows how to combine the driver VIs for these tasks.

Input/output settings

Conn. Axes: 0, A
AI channels: 0, Dev1/ai0
AO channels: 0, Dev1/ao0
Lower voltage limit (V): 0,000000
Upper voltage limit (V): 10,00000
AI Sample Rate: 1000,00
AI Samples/Channel: 1000

error in: status: no error; code: 0; source:

Configuration_Setup is running

Conn. Axis: A
+/- movement range (V): 5,000000
Voltage: 5,012207

error out: status: no error; code: 0; source:

3.2. Analog_Configuration_Setup.vi

This VI performs a fully automatic initialization of the selected system (task and global settings) in the LabVIEW environment. Use the Help - Show Context Help menu sequence in the LabVIEW environment to display the Context Help window with control/indicator descriptions.

After successful run of this VI all command VIs are ready to use. First specify the following:

- System number (= 1 in a one-system-only configuration),
- Input/output settings
- Timeout value (in milliseconds)
- HyperBit settings (HyperBit password required).
  - The Bit rate should be at least double the resonant frequency of the stages connected. If you have different types of stages connected, select the Bit rate to be at least double the maximum resonant frequencies of all stages.
  - The PWM rate must be at least double the Bit rate. If you experience performance problems due to too much bus traffic, decrease the PWM rate value.
  - HB StayOnPos: Select if the position reached with HyperBit ON is to hold. If this control is TRUE and HyperBit is switched ON, LabVIEW will continue to output a hyperbitted target position after the axes have reached their target position. In that way position values which do not correspond to the physical bit resolution of the DAC card can be hold. If this control is FALSE, the analog output will stop after having reached the target position and if the target position does not correspond to the physical bit resolution of the DAC board, the resulting final position will be not as exact. If HyperBit is switched OFF, this control does not have any effect.
- Conversion factor settings.
- Whether the velocity control mode is to be switched ON.
- If velocity control mode is selected to be switched ON, the velocity to set.
- Whether or not all axes are connected and can be moved
- If not, which axes can be moved
- Whether all axes are to be moved to the mid-positions of their travel ranges.
- Whether an offset and gain correction is to be performed by moving the given axes to their minimum and maximum position values.

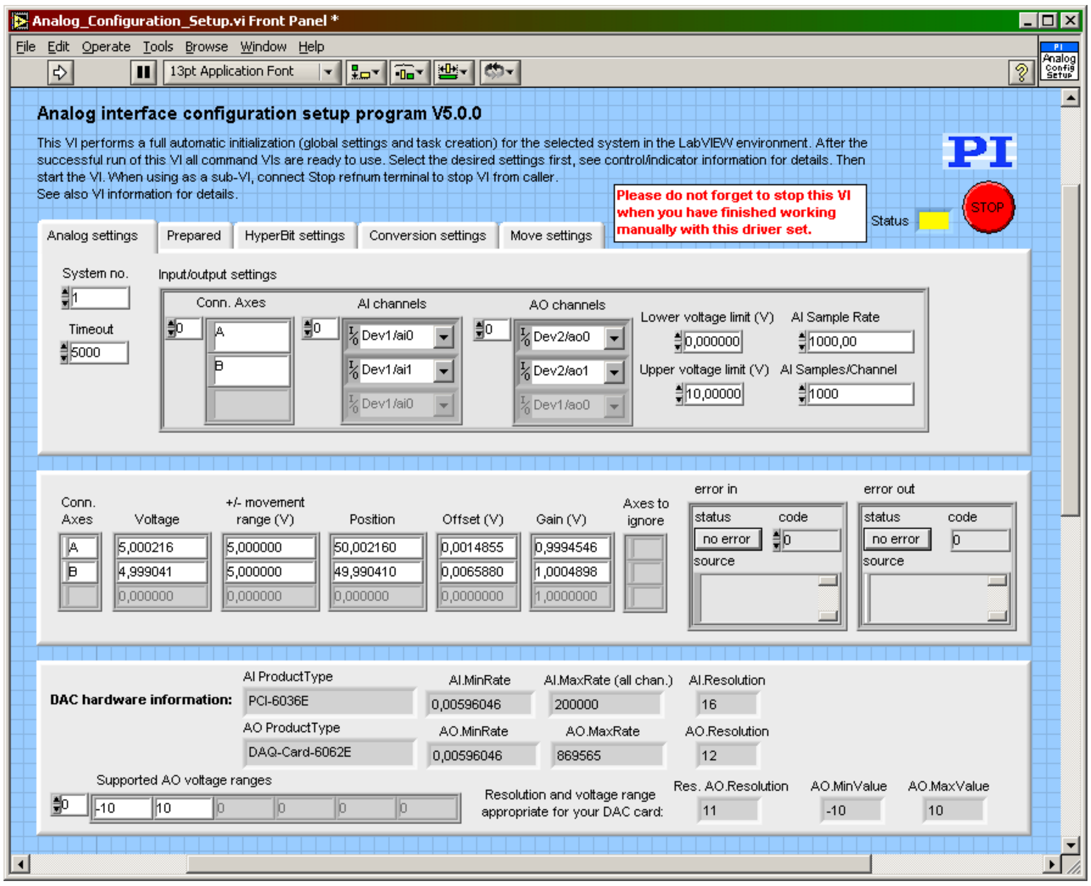
If you run “Analog_Configuration_Setup.vi” manually (and have not called the VI from another VI) because you want to work with the VIs of this driver set manually, the VI will automatically stay idle when all tasks are finished and you can run all command VIs separately afterwards. Do not forget to press the STOP button when you have finished working manually with this driver set.

Now start the VI.

"Analog_Configuration_Setup.vi" performs the following initialization tasks:

1. Defines the analog input task. Determines AI ProductType, AI Resolution, AI min. and max. rate and checks if selected AI Rate is supported by DAC board for given number of connected axes.
2. Defines the analog output task. Determines AO ProductType, supported AO voltage ranges, AO max. and min. rate and AO resolution and calculates, which AO voltage range and AO offset (if supported) is best for the chosen lower and upper voltage limits to achieve the highest possible resolution value. It sets the AO voltage range and offset (if supported) accordingly and calculates the resulting AO resolution for the required voltage working range.
3. Defines the selected system to be "Analog"
4. Runs "Define connected axes.vi" with Read from controller = FALSE and Invert Order? = FALSE.
5. Initializes all Global variables and dynamic VIs.
6. Writes the given settings to the corresponding Global variables.
7. Performs an offset and gain correction, if appropriate, and writes the calculated gain and offset values to Global Analog.vi.
8. Reads the voltage of all axes (VOL?).
9. If Move all to middle? is TRUE , moves each axis to the middle position of its range (MOV). During motion the current voltage and position values are displayed on the front panel.
10. If Move all to middle? is FALSE, moves only axes which are set to TRUE in the corresponding boolean field of Axes to move to the middle position of their range (MOV).
11. Writes the HyperBit settings to Global Analog.vi (HyperBit is inactive up to this point).
12. Translate an error (if any) which corresponds with a GCS error number to Source from Error out.

Depending on the call chain of "Analog_Configuration_Setup.vi" the VI will now either stop (correct behavior when called from another VI) or it will remain idle (correct behavior when command VIs from this driver set are to run manually, i.e. non-programmatically). In the latter case do not forget to press the STOP button when you have finished working with the command VIs.


Analog_Configuration_Setup.vi Front Panel *
File Edit Operate Tools Browse Window Help
13pt Application Font
STOP

Analog interface configuration setup program V5.0.0

This VI performs a full automatic initialization (global settings and task creation) for the selected system in the LabVIEW environment. After the successful run of this VI all command VIs are ready to use. Select the desired settings first, see control/indicator information for details. Then start the VI. When using as a sub-VI, connect Stop refnum terminal to stop VI from caller.
See also VI information for details.

Please do not forget to stop this VI when you have finished working manually with this driver set.

Status

Analog settings | Prepared | HyperBit settings | Conversion settings | Move settings

System no.: 1
Timeout: 5000

Input/output settings

Conn. Axes	AI channels	AO channels
A	Dev1/ai0	Dev2/ao0
B	Dev1/ai1	Dev2/ao1
	Dev1/ai0	Dev1/ao0

Lower voltage limit (V): 0,000000
Upper voltage limit (V): 10,00000
AI Sample Rate: 1000,00
AI Samples/Channel: 1000

Conn. Axes	Voltage	+/- movement range (V)	Position	Offset (V)	Gain (V)	Axes to ignore
A	5,000216	5,000000	50,002160	0,0014855	0,9994546	
B	4,999041	5,000000	49,990410	0,0065880	1,0004898	
	0,000000	0,000000	0,000000	0,0000000	1,0000000	

error in: status no error, code 0, source
error out: status no error, code 0, source

DAC hardware information:

AI ProductType	AI.MinRate	AI.MaxRate (all chan.)	AI.Resolution
PCI-6036E	0,00596046	200000	16

AO ProductType	AO.MinRate	AO.MaxRate	AO.Resolution
DAQ-Card-6062E	0,00596046	869565	12

Supported AO voltage ranges: 0 | -10 | 10 | 0 | 0 | 0 | 0

Resolution and voltage range appropriate for your DAC card:

Res. AO.Resolution	AO.MinValue	AO.MaxValue
11	-10	10

If you generate too many HyperBits by setting the PWM rate too high, the rate capacity of the card or the load capacity of the computer's bus can be exceeded. This may result in a runtime error, or sluggish operation during HyperBit analog output. Solutions: set your VIs to request a lower PWM rate; check your NIMax configuration settings to ensure DMA is being used; or use a faster card. Keep in mind that DAC granularity is only one of many possible limiters to system resolution. Ambient noise and vibration and electronic/amplifier noise, for example, also limit what your system can achieve. There is no point generating HyperBits below the fundamental noise floor.

Use this VI as the initialization VI for any analog system in your application.

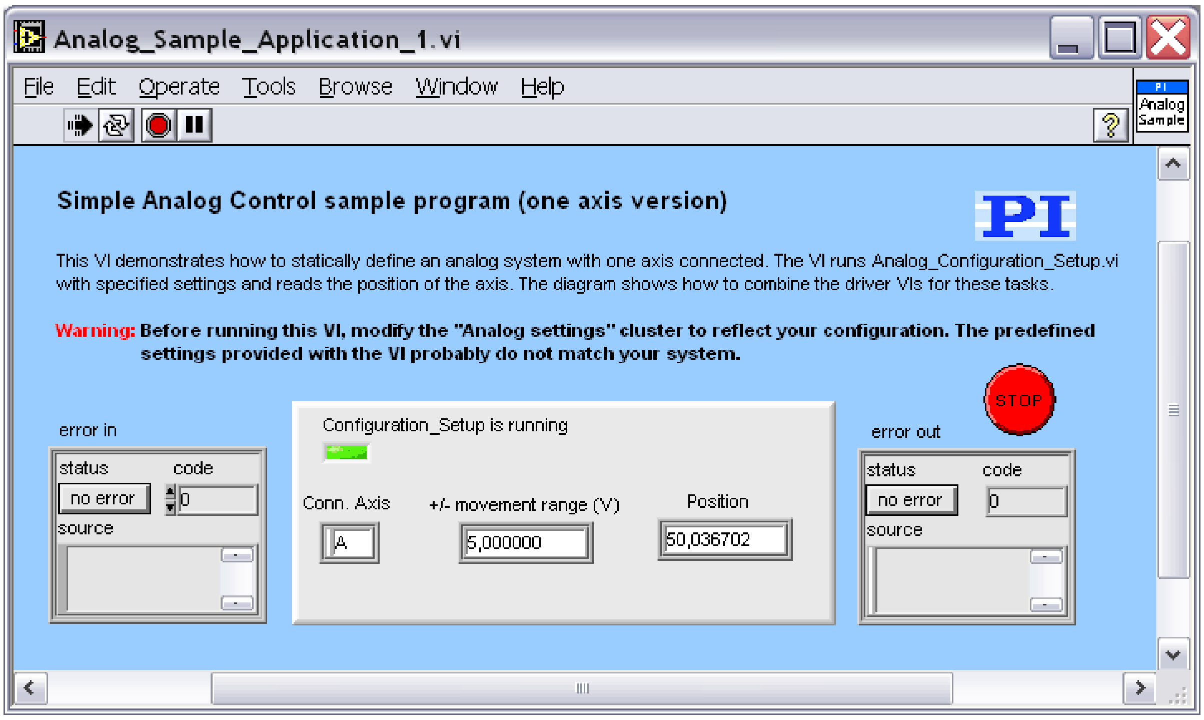
When using as a sub-VI, connect Stop refnum terminal to stop VI from caller.

As the initialization is a complex procedure which uses a large number of sub-VIs, Analog_Configuration_Setup.vi is password-protected, meaning that you cannot see or modify the diagram. In this way, the full initialization is packed into one single and fully tested procedure which you simply insert into your own application program. For security reasons as well as your convenience, we recommend that you not modify this VI.

3.3. Analog_Sample_Application_1.vi

This VI demonstrates how to statically define an analog system with one axis connected. The VI runs Analog_Configuration_Setup.vi with specified settings and reads the position of the axis. The diagram shows how to combine the driver VIs for these tasks.

Warning: Before running this VI, modify the Analog settings cluster on the block diagram to reflect your configuration. The predefined settings provided with the VI probably do not match your system.



Analog_Sample_Application_1.vi

Simple Analog Control sample program (one axis version)

This VI demonstrates how to statically define an analog system with one axis connected. The VI runs Analog_Configuration_Setup.vi with specified settings and reads the position of the axis. The diagram shows how to combine the driver VIs for these tasks.

Warning: Before running this VI, modify the "Analog settings" cluster to reflect your configuration. The predefined settings provided with the VI probably do not match your system.

error in: status no error, code 0, source

Configuration_Setup is running

Conn. Axis: A

+/- movement range (V): 5,000000

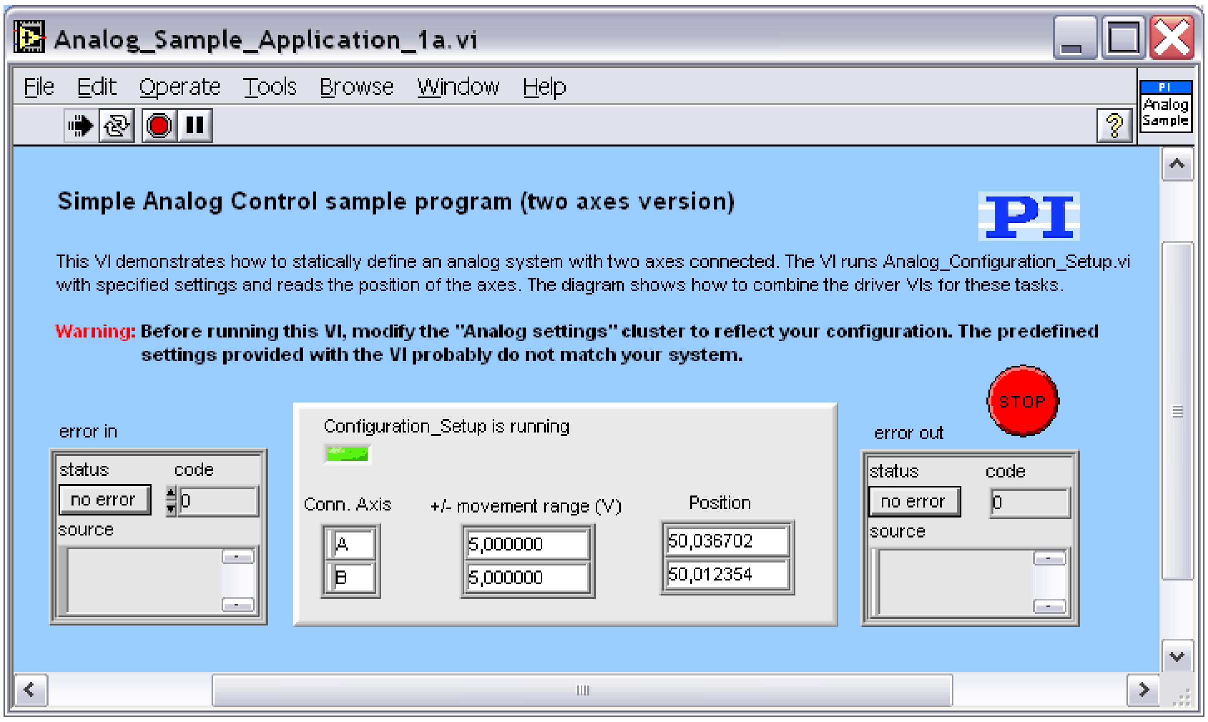
Position: 50,036702

error out: status no error, code 0, source

3.4. Analog_Sample_Application_1a.vi

This VI demonstrates how to statically define an analog system with two axes connected. The VI runs Analog_Configuration_Setup.vi with specified settings and reads the position of the axes. The diagram shows how to combine the driver VIs for these tasks.

Warning: Before running this VI, modify the Analog settings cluster on the block diagram to reflect your configuration. The predefined settings provided with the VI probably do not match your system.


Analog_Sample_Application_1a.vi
File Edit Operate Tools Browse Window Help
PI Analog Sample
STOP

Simple Analog Control sample program (two axes version)

This VI demonstrates how to statically define an analog system with two axes connected. The VI runs Analog_Configuration_Setup.vi with specified settings and reads the position of the axes. The diagram shows how to combine the driver VIs for these tasks.

Warning: Before running this VI, modify the "Analog settings" cluster to reflect your configuration. The predefined settings provided with the VI probably do not match your system.

error in: status no error, code 0, source

Configuration_Setup is running

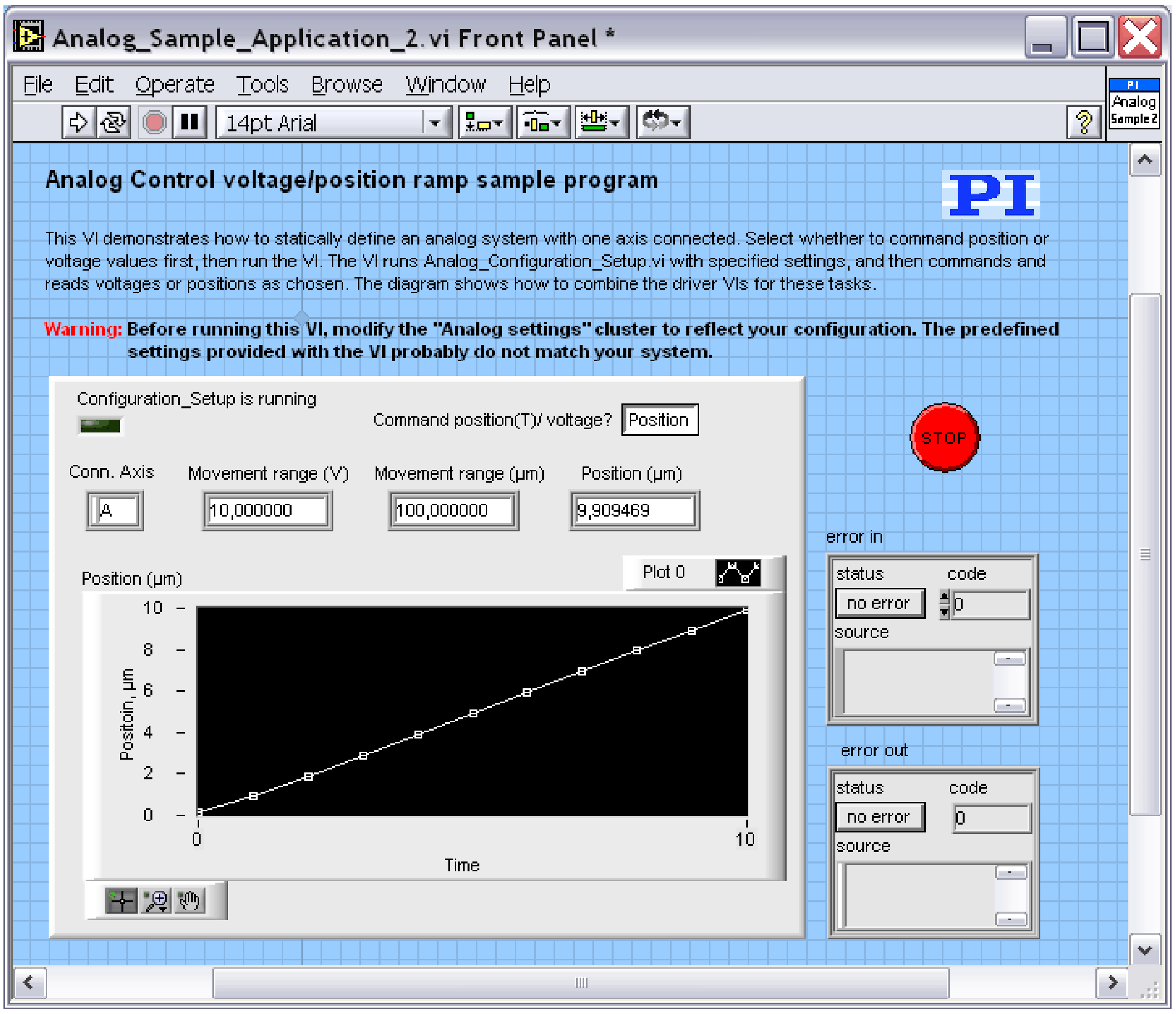
Conn. Axis	+/- movement range (V)	Position
A	5,000000	50,036702
B	5,000000	50,012354

error out: status no error, code 0, source

3.5. Analog_Sample_Application_2.vi

This VI demonstrates how to statically define an analog system with one axis connected. Select whether to command position or voltage values first, then run the VI. The VI runs Analog_Configuration_Setup.vi with specified settings, and then commands and reads voltages or positions as chosen. The diagram shows how to combine the driver VIs for these tasks.

Warning: Before running this VI, modify the Analog settings cluster on the block diagram to reflect your configuration. The predefined settings provided with the VI probably do not match your system.


Analog_Sample_Application_2.vi Front Panel *
File Edit Operate Tools Browse Window Help
14pt Arial
STOP

Analog Control voltage/position ramp sample program

This VI demonstrates how to statically define an analog system with one axis connected. Select whether to command position or voltage values first, then run the VI. The VI runs Analog_Configuration_Setup.vi with specified settings, and then commands and reads voltages or positions as chosen. The diagram shows how to combine the driver VIs for these tasks.

Warning: **Before running this VI, modify the "Analog settings" cluster to reflect your configuration. The predefined settings provided with the VI probably do not match your system.**

Configuration_Setup is running

Command position(T)/ voltage? Position

Conn. Axis	Movement range (V)	Movement range (µm)	Position (µm)
A	10,000000	100,000000	9,909469

Position (µm) Plot 0

Positoin, µm 10 8 6 4 2 0

0 10 Time

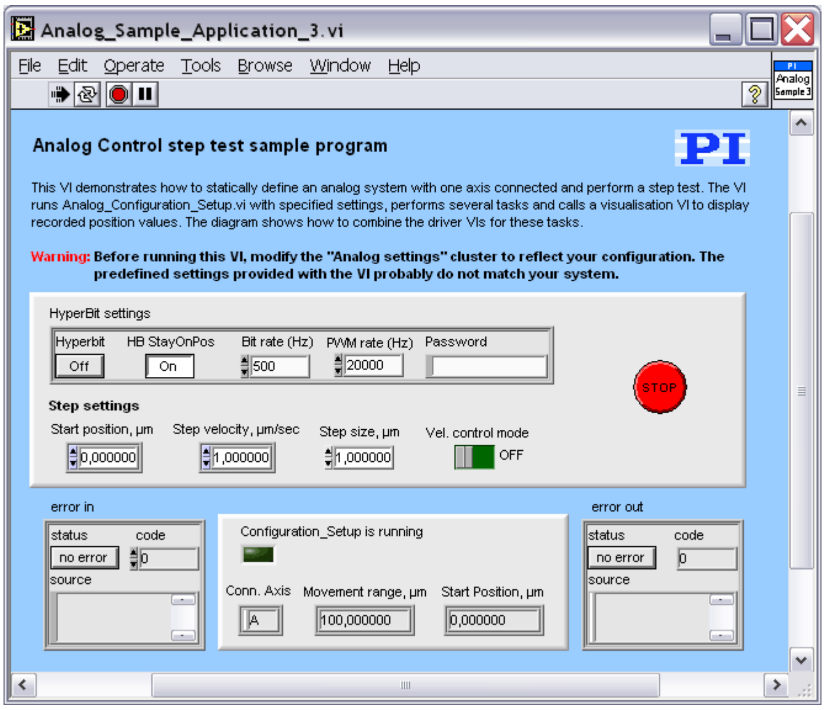
error in: status no error, code 0, source

error out: status no error, code 0, source

3.6. Analog_Sample_Application_3.vi

This VI This VI demonstrates how to statically define an analog system with one axis connected and perform a step test. The VI runs Analog_Configuration_ Setup.vi with specified settings, performs several tasks and calls a visualisation VI to display recorded position values. The diagram shows how to combine the driver VIs for these tasks.

Warning: Before running this VI, modify the Analog settings cluster on the block diagram to reflect your configuration. The predefined settings provided with the VI probably do not match your system.


Analog_Sample_Application_3.vi
File Edit Operate Tools Browse Window Help
Analog Sample 3
PI
STOP

Analog Control step test sample program

This VI demonstrates how to statically define an analog system with one axis connected and perform a step test. The VI runs Analog_Configuration_Setup.vi with specified settings, performs several tasks and calls a visualisation VI to display recorded position values. The diagram shows how to combine the driver VIs for these tasks.

Warning: Before running this VI, modify the "Analog settings" cluster to reflect your configuration. The predefined settings provided with the VI probably do not match your system.

HyperBit settings

Hyperbit	HB StayOnPos	Bit rate (Hz)	PWM rate (Hz)	Password
Off	On	500	20000	

Step settings

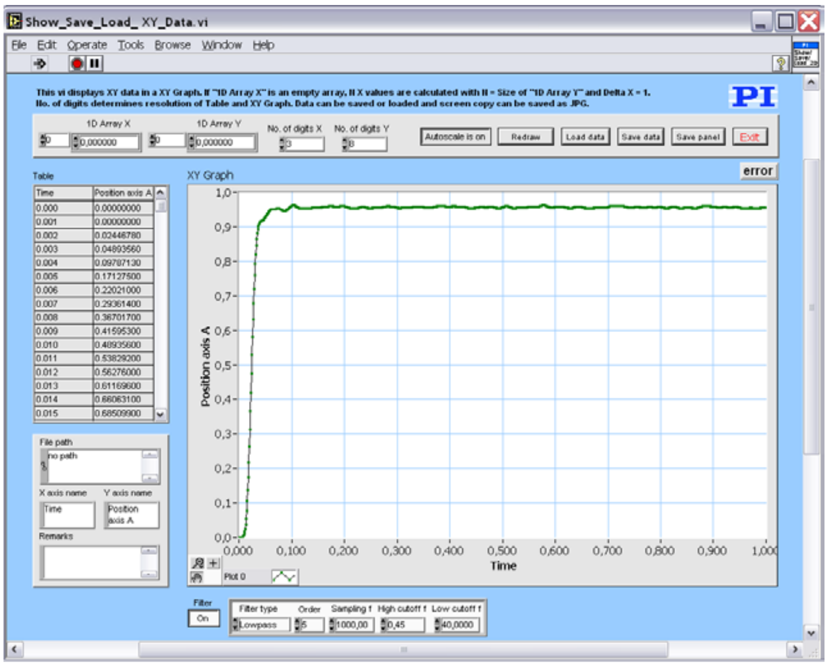
Start position, µm	Step velocity, µm/sec	Step size, µm	Vel. control mode
0,000000	1,000000	1,000000	OFF

error in: status no error; code 0; source

Configuration_Setup is running

Conn. Axis	Movement range, µm	Start Position, µm
A	100,000000	0,000000

error out: status no error; code 0; source



This vi displays XY data in a XY Graph. If "1D Array X" is an empty array, N X values are calculated with N = Size of "1D Array Y" and Delta X = 1. No. of digits determines resolution of Table and XY Graph. Data can be saved or loaded and screen copy can be saved as JPG.

1D Array X: 0, 0,000000 — 1D Array Y: 0, 0,000000 — No. of digits X: 3 — No. of digits Y: 8

Table

Time	Position axis A
0.000	0.00000000
0.001	0.00000000
0.002	0.02446780
0.003	0.04893560
0.004	0.09787130
0.005	0.17127500
0.006	0.22021000
0.007	0.29361400
0.008	0.36701700
0.009	0.41595300
0.010	0.48935600
0.011	0.53829200
0.012	0.56276000
0.013	0.61169600
0.014	0.66063100
0.015	0.68509900

File path: no path

X axis name: Time — Y axis name: Position axis A

Remarks

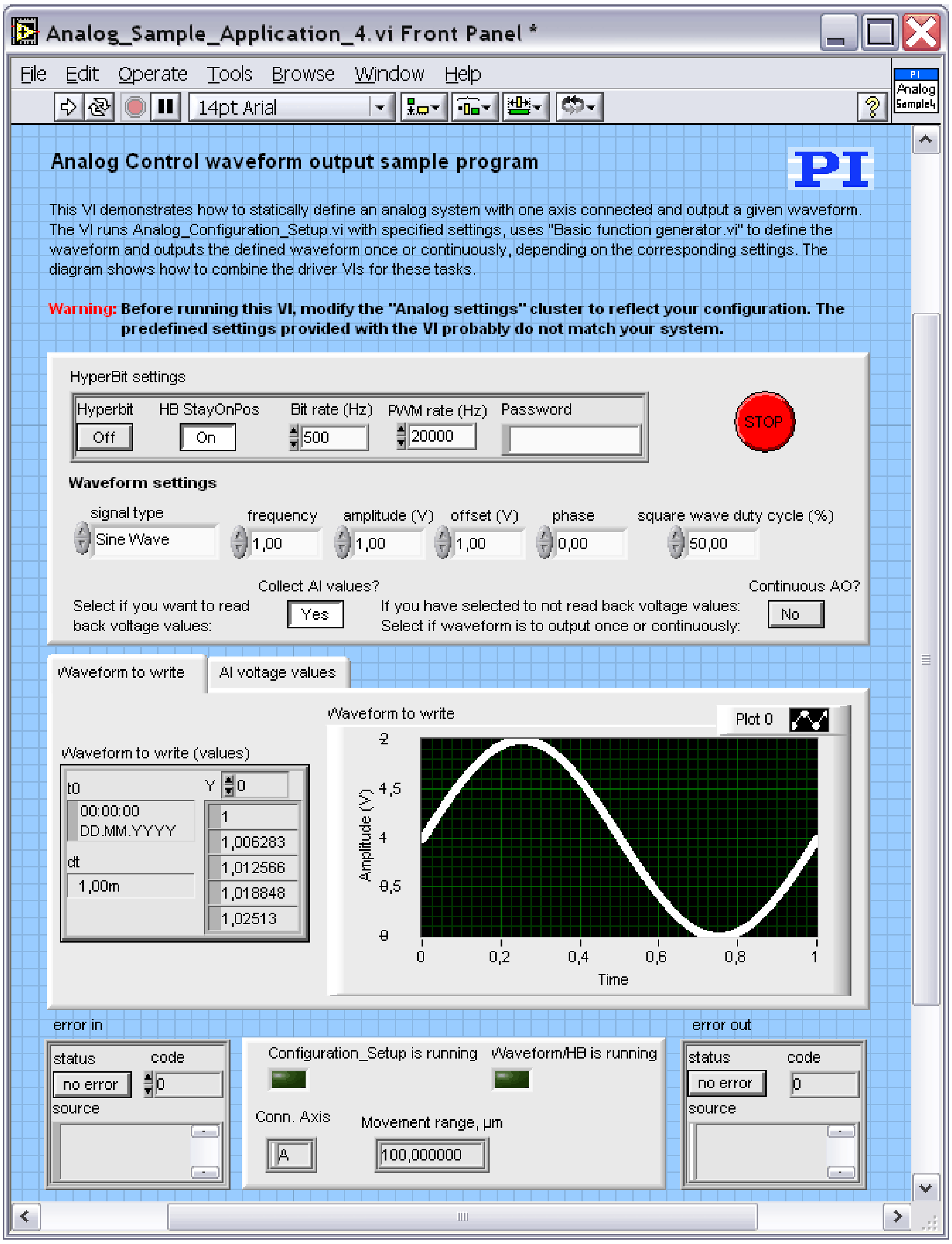
Filter: On

Filter type	Order	Sampling f	High cutoff f	Low cutoff f
Lowpass	5	1000,00	0,45	40,0000

3.7. Analog_Sample_Application_4.vi

This VI demonstrates how to statically define an analog system with one axis connected and output a given waveform. The VI runs Analog_Configuration_Setup.vi with specified settings, uses "Basic function generator.vi" to define the waveform and outputs the defined waveform once or continuously, depending on the corresponding settings. Optionally it collects AI voltage values while outputting AO values. The diagram shows how to combine the driver VIs for these tasks.

Warning: Before running this VI, modify the Analog settings cluster on the block diagram to reflect your configuration. The predefined settings provided with the VI probably do not match your system.



Analog Control waveform output sample program

This VI demonstrates how to statically define an analog system with one axis connected and output a given waveform. The VI runs Analog_Configuration_Setup.vi with specified settings, uses "Basic function generator.vi" to define the waveform and outputs the defined waveform once or continuously, depending on the corresponding settings. The diagram shows how to combine the driver VIs for these tasks.

Warning: Before running this VI, modify the "Analog settings" cluster to reflect your configuration. The predefined settings provided with the VI probably do not match your system.

HyperBit settings

Hyperbit	HB StayOnPos	Bit rate (Hz)	PWM rate (Hz)	Password
Off	On	500	20000	

Waveform settings

signal type	frequency	amplitude (V)	offset (V)	phase	square wave duty cycle (%)
Sine Wave	1,00	1,00	1,00	0,00	50,00

Collect AI values? Select if you want to read back voltage values: Yes

Continuous AO? If you have selected to not read back voltage values: Select if waveform is to output once or continuously: No

Waveform to write (values)

t0: 00:00:00 DD.MM.YYYY
dt: 1,00m
Y: 0
1
1,006283
1,012566
1,018848
1,02513

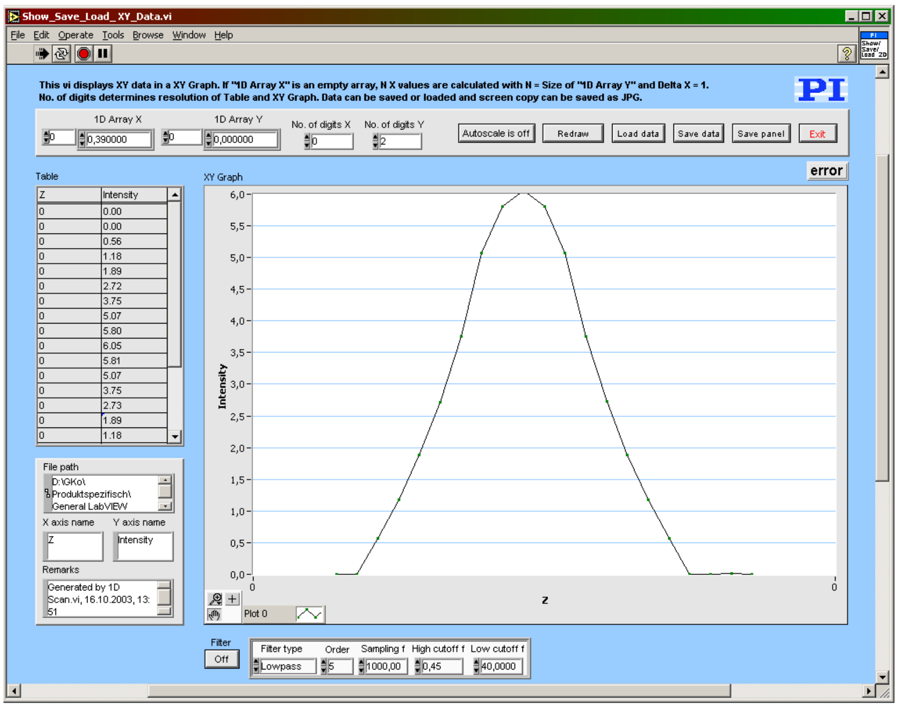
error in: status: no error, code: 0, source:

Configuration_Setup is running
Waveform/HB is running
Conn. Axis: A
Movement range, µm: 100,000000

error out: status: no error, code: 0, source:

3.8. Show_Save_Load_XY_Data.vi

This VI displays XY data in an XY Graph. If 1D Array X is an empty array, N X values are calculated with N = Size of 1D Array Y and Delta X = 1. No. of digits determines the resolution of Table and XY Graph. Data can be saved or loaded and a screen copy can be saved as JPG.


Show_Save_Load_XY_Data.vi

This vi displays XY data in a XY Graph. If "1D Array X" is an empty array, N X values are calculated with N = Size of "1D Array Y" and Delta X = 1.
No. of digits determines resolution of Table and XY Graph. Data can be saved or loaded and screen copy can be saved as JPG.

1D Array X: 0 | 0,390000
1D Array Y: 0 | 0,000000
No. of digits X: 0
No. of digits Y: 2

Table

Z	Intensity
0	0.00
0	0.00
0	0.56
0	1.18
0	1.89
0	2.72
0	3.75
0	5.07
0	5.80
0	6.05
0	5.81
0	5.07
0	3.75
0	2.73
0	1.89
0	1.18

File path: D:\GKo\Produktspezifisch\General LabVIEW
X axis name: Z
Y axis name: Intensity
Remarks: Generated by 1D Scan.vi, 16.10.2003, 13:51

Filter: Off

Filter type	Order	Sampling f	High cutoff f	Low cutoff f
Lowpass	5	1000,00	0,45	40,0000

If data (1D Array X, 1D Array Y) are sent to the VI via the corresponding connectors, the VI will display the corresponding graphics after being called. To load data at runtime, press the Load data button. A dialog will pop up where a data file to open can be selected. The VI can read data in GCSArray, GCSTable and simple ASCII column format. Autoscale can be switched on or off. If Autoscale is off, the Y axis of the graph is scaled from 0-10.

Filter can be used to apply a filter to the current graph. For Filter = TRUE, a Lowpass, Highpass, Bandpass or Bandstop filter with appropriate settings can be selected.

Press Save data to save data (file header and numerical data). Data will be saved in GCS Array format. The file header will contain information given in X axis name, Y axis name and Remarks. With Save panel a screen copy of this VI can be saved as a JPG file. XY Graph will show the Y values over the corresponding X values. Table contains the numerical values for X and Y. Press Exit to stop execution of this VI.

Valid for	Analog systems, C-866, C-880, E-753, E-755, E-761, F-206, M-8X0
Input	1D Array X (empty num. array), 1D Array Y (empty num. array), 2D Array Z (empty 2D num. array), No. of digits X (, No. of digits Y, No. of digits Z, Autoscale, Error in (no error)
Output	Error out
Remarks	-

4. PI Systems Currently Supported by This Driver Set

Product	works with LabVIEW driver version (or higher)	if product firmware/ drivers version is equal to or newer than
Analog	5.2.2	-
C-702	4.0.0	1.4.0
C-843	2.01 – 2.02 2.05 – 2.06 3.1.2., 3.1.2a 3.4.3 3.6.1	MC-DLL 1.0.2.2 MC-DLL 1.0.2.3 MC-DLL 1.0.2.3 MC-DLL 1.0.2.8 MC-DLL 1.0.2.8 GCS_DLL 1.3.1
C-843.PM	3.1.0 3.4.3a 3.6.2	MC-DLL 1.0.2.5 MC-DLL 1.0.2.5 MC-DLL 1.0.2.5 GCS_DLL 1.3.0
C-848	3.0.2	1.0
C-865	3.3.0	MC_C865.dll 1.0
C-866	5.2.1	MC_C866.dll 1.0
C-880	1.1 1.2 2.04 2.05 – 2.06 3.2.0	2.00 2.10 2.20 2.21 2.40
C-880K005	2.06	1.0
C-880K006	2.06	1.0
C-880K007	2.06	1.0
E-516	1.0 – 2.02 2.05 – 2.06 3.4.2	DSP V3.01, MCU V5 DSP V3.11, MCU V5 DSP V3.30, MCU V5
E-710 3- & 4-channel versions	3.4.0 3.4.4 (a, b)	5.027 5.0.33, 6.0.33
E-710 6-channel	3.4.4 (a, b)	2.13
E-753	5.2.0	1.0.0
E-755	5.1.0	2.0.4.1 E7XX_GCS2_DLL.dll V1.1.0
E-761	3.5.0	1.0.0
E-816	2.01 – 2.06	2.02
F-206	1.1 – 2.06	Fhx0035

M-840	2.03 – 2.06	Hex0037
	2.2.0	Hex0045
	3.0.1	Hex0050
	3.1.1	Hex0051
M-850	2.03 – 2.06	Hex0040
	3.0.1	Hex0050
	3.1.1	Hex0051
Mercury™	3.6.0	1.0.6 PI_MERCURY_GCS_DLL.dll V 1.0.0.17

5. Appendix A

Error codes are not unambiguous, but can result from a PI error message or LabVIEW internal error code. In addition to the list below see National Instruments error codes.

100	PI LabVIEW driver reports error. See source control for details.
0	No error
1	Parameter syntax error
2	Unknown command
3	Command length out of limits or command buffer overrun
4	Error while scanning
5	Unallowable move attempted on unreferenced axis, or move attempted with servo off
6	Parameter for SGA not valid
7	Position out of limits
8	Velocity out of limits
9	Attempt to set pivot point while U,V and W not all 0
10	Controller was stopped by command
11	Parameter for SST or for one of the embedded scan algorithms out of range
12	Invalid axis combination for fast scan
13	Parameter for NAV out of range
14	Invalid analog channel
15	Invalid axis identifier
16	Unknown stage name
17	Parameter out of range
18	Invalid macro name
19	Error while recording macro
20	Macro not found
21	Axis has no brake
22	Axis identifier specified more than once
23	Illegal axis
24	Incorrect number of parameters
25	Invalid floating point number
26	Parameter missing
27	Soft limit out of range
28	No manual pad found

29	No more step-response values
30	No step-response values recorded
31	Axis has no reference sensor
32	Axis has no limit switch
33	No relay card installed
34	Command not allowed for selected stage(s)
35	No digital input installed
36	No digital output configured
37	No more MCM responses
38	No MCM values recorded
39	Controller number invalid
40	No joystick configured
41	Invalid axis for electronic gearing, axis can not be slave
42	Position of slave axis is out of range
43	Slave axis cannot be commanded directly when electronic gearing is enabled
44	Calibration of joystick failed
45	Referencing failed
46	OPM (Optical Power Meter) missing
47	OPM (Optical Power Meter) not initialized or cannot be initialized
48	OPM (Optical Power Meter) Communication Error
49	Move to limit switch failed
50	Attempt to reference axis with referencing disabled
51	Selected axis is controlled by joystick
52	Controller detected communication error
53	MOV! motion still in progress
54	Unknown parameter
55	No commands were recorded with REP
56	Password invalid
57	Data Record Table does not exist
58	Source does not exist; number too low or too high
59	Source Record Table number too low or too high
60	Protected Param: current Command Level (CCL) too low
61	Command execution not possible while Autozero is running
62	Autozero requires at least one linear axis
63	Initialization still in progress
64	Parameter is read-only
65	Parameter not found in non-volatile memory
66	Voltage out of limits

67	Not enough memory available for requested wave curve
68	Not enough memory available for DDL table; DDL can not be started
69	Time delay larger than DDL table; DDL can not be started
70	The requested arrays have different lengths; query them separately
71	Attempt to restart the generator while it is running in single step mode
72	Motion commands and wave generator activation are not allowed when analog target is active
73	Motion commands are not allowed when wave generator output is active; use WGO to disable generator output
74	No sensor channel or no piezo channel connected to selected axis (sensor and piezo matrix)
75	Generator started (WGO) without having selected a wave table (WSL).
76	Interface buffer did overrun and command couldn't be received correctly
77	Data Record Table does not hold enough recorded data
78	Data Record Table is not configured for recording
100	PI LabVIEW driver reports error. See source control for details.
200	No stage connected to axis
201	File with axis parameters not found
202	Invalid axis parameter file
203	Backup file with axis parameters not found
204	PI internal error code 204
205	SMO with servo on
206	uudecode: incomplete header
207	uudecode: nothing to decode
208	uudecode: illegal UUE format
209	CRC32 error
210	Illegal file name (must be 8-0 format)
211	File not found on controller
212	Error writing file on controller
213	VEL command not allowed in DTR Command Mode
214	Position calculations failed
215	The connection between controller and stage may be broken
216	The connected stage has driven into a limit switch, call CLR to resume operation
217	Strut test command failed because of an unexpected strut stop
218	While MOV! is running position can only be estimated!
219	Position was calculated during MOV motion
301	Send buffer overflow
302	Voltage out of limits

303	Attempt to set voltage when servo on
304	Received command is too long
305	Error while reading/writing EEPROM
306	Error on I2C bus
307	Timeout while receiving command
308	A lengthy operation has not finished in the expected time
309	Insufficient space to store macro
310	Configuration data has old version number
311	Invalid configuration data
333	Internal hardware error
555	BasMac: unknown controller error
601	Not enough memory
602	Hardware voltage error
603	Hardware temperature out of range
1000	Too many nested macros
1001	Macro already defined
1002	Macro recording not activated
1003	Invalid parameter for MAC
1004	Deleting macro failed
2000	Controller already has a serial number
4000	Sector erase failed
4001	Flash program failed
4002	Flash read failed
4003	HW match code missing/invalid
4004	FW match code missing/invalid
4005	HW version missing/invalid
4006	FW version missing/invalid
4007	FW update failed
0	No error occurred during function call
-1	Error during com operation (could not be specified)
-2	Error while sending data
-3	Error while receiving data
-4	Not connected (no port with given ID open)
-5	Buffer overflow
-6	Error while opening port
-7	Timeout error
-8	There are more lines waiting in buffer
-9	There is no interface or DLL handle with the given ID

-10	Event/message for notification could not be opened
-11	Function not supported by this interface type
-12	Error while sending "echoed" data
-13	IEEE488: System error
-14	IEEE488: Function requires GPIB board to be CIC
-15	IEEE488: Write function detected no listeners
-16	IEEE488: Interface board not addressed correctly
-17	IEEE488: Invalid argument to function call
-18	IEEE488: Function requires GPIB board to be SAC
-19	IEEE488: I/O operation aborted
-20	IEEE488: Interface board not found
-21	IEEE488: Error performing DMA
-22	IEEE488: I/O operation started before previous operation completed
-23	IEEE488: No capability for intended operation
-24	IEEE488: File system operation error
-25	IEEE488: Command error during device call
-26	IEEE488: Serial poll-status byte lost
-27	IEEE488: SRQ remains asserted
-28	IEEE488: Return buffer full
-29	IEEE488: Address or board locked
-30	RS-232: 5 data bits with 2 stop bits is an invalid combination, as is 6, 7, or 8 data bits with 1.5 stop bits
-31	RS-232: Error configuring the COM port
-32	Error dealing with internal system resources (events, threads, ...)
-33	A DLL or one of the required functions could not be loaded
-34	FTDIUSB: invalid handle
-35	FTDIUSB: device not found
-36	FTDIUSB: device not opened
-37	FTDIUSB: IO error
-38	FTDIUSB: insufficient resources
-39	FTDIUSB: invalid parameter
-40	FTDIUSB: invalid baud rate
-41	FTDIUSB: device not opened for erase
-42	FTDIUSB: device not opened for write
-43	FTDIUSB: failed to write device
-44	FTDIUSB: EEPROM read failed
-45	FTDIUSB: EEPROM write failed
-46	FTDIUSB: EEPROM erase failed

-47	FTDIUSB: EEPROM not present
-48	FTDIUSB: EEPROM not programmed
-49	FTDIUSB: invalid arguments
-50	FTDIUSB: not supported
-51	FTDIUSB: other error
-52	Error while opening the COM port: was already open
-53	Checksum error in received data from COM port
-54	Socket not ready, you should call the function again
-55	Port is used by another socket
-56	Socket not connected (or not valid)
-57	Connection terminated (by peer)
-58	Can't connect to peer
-59	Operation was interrupted by a nonblocked signal
-1001	Unknown axis identifier
-1002	Number for NAV out of range--must be in [1,10000]
-1003	Invalid value for SGA--must be one of 1, 10, 100, 1000
-1004	Controller sent unexpected response
-1005	No manual control pad installed, calls to SMA and related commands are not allowed
-1006	Invalid number for manual control pad knob
-1007	Axis not currently controlled by a manual control pad
-1008	Controller is busy with some lengthy operation (e.g. reference move, fast scan algorithm)
-1009	Internal error--could not start thread
-1010	Controller is (already) in macro mode--command not valid in macro mode
-1011	Controller not in macro mode--command not valid unless macro mode active
-1012	Could not open file to write or read macro
-1013	No macro with given name on controller, or macro is empty
-1014	Internal error in macro editor
-1015	One or more arguments given to function is invalid (empty string, index out of range, ...)
-1016	Axis identifier is already in use by a connected stage
-1017	Invalid axis identifier
-1018	Could not access array data in COM server
-1019	Range of array does not fit the number of parameters
-1020	Invalid parameter ID given to SPA or SPA?
-1021	Number for AVG out of range--must be >0
-1022	Incorrect number of samples given to WAV
-1023	Generation of wave failed

-1024	Motion error while axis in motion, call CLR to resume operation
-1025	Controller is (already) running a macro
-1026	Configuration of PZT stage or amplifier failed
-1027	Current settings are not valid for desired configuration
-1028	Unknown channel identifier
-1029	Error while reading/writing wave generator parameter file
-1030	Could not find description of wave form. Maybe WG.INI is missing?
-1031	The WGWaveEditor DLL function was not found at startup
-1032	The user cancelled a dialog
-1033	Error from C-844 Controller
-1034	DLL necessary to call function not loaded, or function not found in DLL
-1035	The open parameter file is protected and cannot be edited
-1036	There is no parameter file open
-1037	Selected stage does not exist
-1038	There is already a parameter file open. Close it before opening a new file
-1039	Could not open parameter file
-1040	The version of the connected controller is invalid
-1041	Parameter could not be set with SPA--parameter not defined for this controller!
-1042	The maximum number of wave definitions has been exceeded
-1043	The maximum number of wave generators has been exceeded
-1044	No wave defined for specified axis
-1045	Wave output to axis already stopped/started
-1046	Not all axes could be referenced
-1047	Could not find parameter set required by frequency relation
-1048	Command ID given to SPP or SPP? is not valid
-1049	A stage name given to CST is not unique
-1050	A uuencoded file transfered did not start with "begin" followed by the proper filename
-1051	Could not create/read file on host PC
-1052	Checksum error when transfering a file to/from the controller
-1053	The PiStages.dat database could not be found. This file is required to connect a stage with the CST command
-1054	No wave being output to specified axis
-1055	Invalid password
-1056	Error during communication with OPM (Optical Power Meter), maybe no OPM connected
-1057	WaveEditor: Error during wave creation, incorrect number of parameters
-1058	WaveEditor: Frequency out of range
-1059	WaveEditor: Error during wave creation, incorrect index for integer parameter

-1060	WaveEditor: Error during wave creation, incorrect index for floating point parameter
-1061	WaveEditor: Error during wave creation, could not calculate value
-1062	WaveEditor: Graph display component not installed
-1063	User Profile Mode: Command is not allowed, check for required preparatory commands
-1064	User Profile Mode: First target position in User Profile is too far from current position
-1065	Controller is (already) in User Profile Mode
-1066	User Profile Mode: Block or Data Set index out of allowed range
-1067	ProfileGenerator: No profile has been created yet
-1068	ProfileGenerator: Generated profile exceeds limits of one or both axes
-1069	ProfileGenerator: Unknown parameter ID in Set/Get Parameter command
-1070	ProfileGenerator: Parameter out of allowed range
-1071	User Profile Mode: Out of memory
-1072	User Profile Mode: Cluster is not assigned to this axis
-1073	Unknown cluster identifier
-1074	The installed device driver doesn't match the required version. Please see the documentation to determine the required device driver version.
-1075	The library used doesn't match the required version. Please see the documentation to determine the required library version.
-1076	The interface is currently locked by another function. Please try again later.

6. Index

End of document