Motion VI Library

LabVIEW Programming Manual

Revision 2.1

Precision MicroControl Corporation
2075-N Corte del Nogal
Carlsbad, CA 92009-1415 USA

Tel: (760) 930-0101
Fax: (760) 930-0222

www.pmccorp.com

Information: info@pmccorp.com
Technical Support: support@pmccorp.com

LIMITED WARRANTY

All products manufactured by PRECISION MICROCONTROL CORPORATION are guaranteed to be free from defects in material and workmanship, for a period of five years from the date of shipment. Liability is limited to FOB Factory repair, or replacement, of the product. Other products supplied as part of the system carry the warranty of the manufacturer.

PRECISION MICROCONTROL CORPORATION does not assume any liability for improper use or installation or consequential damage.

Information in this document is subject to change without notice.

Precision MicroControl
2075-N Corte del Nogal
Carlsbad, CA 92009-1415

Phone: (760)930-0101
Fax: (760)930-0222
World Wide Web: www.pmccorp.com
Email:
Information: info@pmccorp.com
Technical support: support@pmccorp.com
Sales: sales@pmccorp.com

Table of Contents

User manual revision history

Revision	Date	Description
2.0	3/25/2002	Initial release
2.1	5/7/2002	Updated to reflect Motion VI Library 2.1

Prologue

This manual has been written as a reference manual for the Motion VI Library. However, this is not meant to be the only document you should reference regarding the use of the Motion VI Library. You will find more application specific information on how to use your motion control card in your User's Manual. Although most of the application examples are written in C++ code, the function names correspond to equivalent VIs, and the examples should give you a good deal of insight as to how the VIs should be used.

Also, you will find other valuable information on how to use your motion control card on your **MotionCD**. There, you will find the following information:

- Tutorials (PowerPoint presentations)
  An Introduction to PMC Motion Control
  Installing a PMC Motion Controller (Does not Address PCI bus controllers)
  Introduction to Motion Control Programming with the Motion Control API
  Servo Systems Primer
  Servo Tuning

- PMC AppNOTES – detailed descriptions of specific motion control applications

- PMC TechNOTES – one page technical support documents

- PMC Product catalogs and brochures

Chapter Contents

- First Time Users
- Required Software
- Online Help

Chapter
1

Introduction

First Time Users

If this is your first time using one of PMC's motion control cards, we would like to welcome you to our unique approach to motion control. We would also like to thank you for reading this introduction section. Here we would like to acquaint you with the steps to properly setup the software for a motion control card with minimal confusion and frustration.

Being Engineers ourselves, we know the excitement of playing with new toys. We do not expect you to read the entire manual prior to using our product, no matter how happy that would make us. However, we would be rather pleased if you would take the time to finish reading this chapter to understand some of tools we provide you with to help you reduce the learning curve dramatically. Remember, all of our software is provided at no charge, and upgrades can be found only a click away on our website, www.pmccorp.com.

You should have received a Quick Start Card with your motion controller. This has all the steps for physical installation as well as software installation nicely numbered with pretty pictures that will guide you quickly through the setup process. You may find an electronic copy of this on the **MotionCD**. Please take a moment to review this card if you have not done so already. This will make your life so much easier!

Required Software

Obviously, you will need software to make our product work, but what software do you need? Since this is a manual about the Motion VI Library, you probably already correctly guessed LabVIEW. However, this is only part of the story. Please take a moment to look at figure 1.

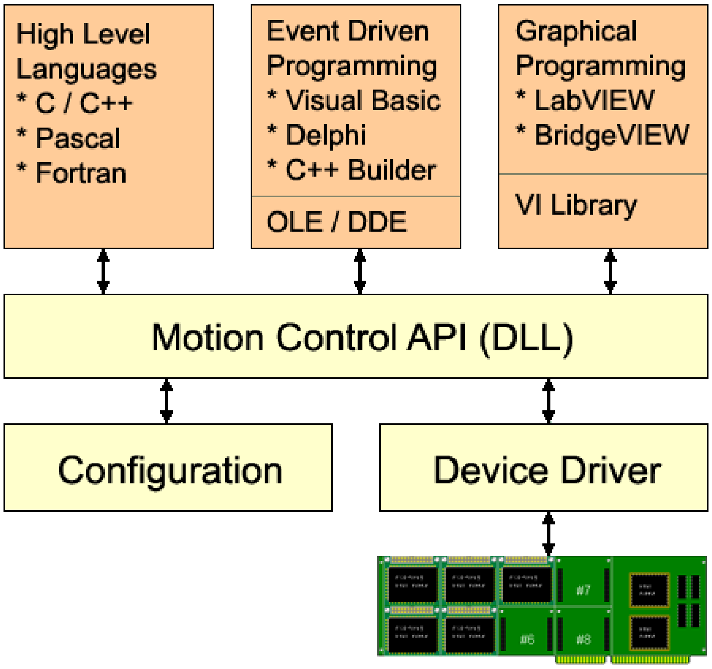


Figure 1: MCAPI and motion control card architectural diagram

You will notice that there are several layers of software between LabVIEW and the motion control card. Each layer provides a level of abstraction which allows the layer above it to be that much simpler. In this way, we can hide low-level details from the programmer, keeping the higher-level code the same across multiple products. The look and feel of the code which you are about to learn will not change from product to product or generation to generation. This approach allows for product developments and enhancements without breaking existing code that our customers have already written.

Figure 1 shows that you will need to install two pieces of software other than LabVIEW. The **Motion Control Application Programming Interface** (MCAPI) includes the low-level device driver and configuration for each of our products. The Motion VI Library contains the familiar VI programming interface.

LabVIEW and the Motion VI Library are only part of what is needed to interface to one of our motion control cards. You will also need to install our **Motion Control API** (MCAPI). When you install the MCAPI, the necessary DLL will also be installed. Without the MCAPI installed, you will not be able to communicate with your motion control card.

The MCAPI should be conveniently installed from the MotionCD that we shipped with the first motion control card, however, you may download the latest MCAPI from our website, www.pmccorp.com. The version numbers between the MCAPI and the Motion VI Library need not match. If you are concerned about the version of the MCAPI that you should install, each VI listing has a category for which version of MCAPI is required. The 2.1 version of the Motion VI Library will require the MCAPI version 2.1c or higher for full functionality.

Install LabVIEW First

Before you install the Motion VI Library you must first install LabVIEW version 5.x or 6.x. This is necessary so that when you install the Motion VI Library its function and control palettes can be added to the LabVIEW menu system, and the online help is placed where LabVIEW can locate it.

When you install the Motion VI Library, please verify that you install into the root of the LabVIEW directory for your installation. The InstallShield will select the proper default directory even if you choose a custom installation. However, if you choose a custom installation, you must be careful not to alter this directory, or you will not have easy access to the installed components. For instance, LabVIEW 5.0 has the following default directory path for installation.

```
C:\Program Files\National Instruments\LabVIEW
```

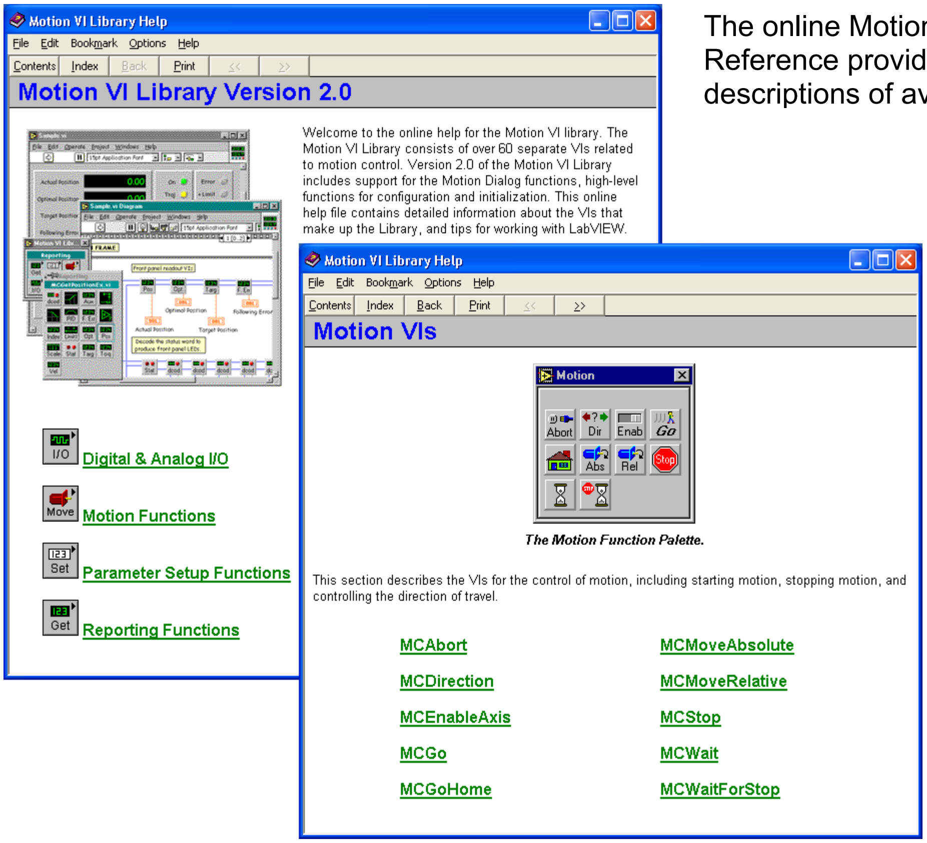
LabVIEW 6.1 will install into the following directory.

```
C:\Program Files\National Instruments\LabVIEW 6.1
```

Online Help

Although this manual includes most of the information found in the Motion VI Library Help file, the online version will be a quicker method of understanding a function when you are sitting at your computer. When the Motion VI Library is properly installed, this help file will be seamlessly integrated into LabVIEW's online help. By right clicking on a Motion VI in the diagram window, you may select Online Help from the menu to bring up the appropriate page on the Motion VI in question. You may also view this help at anytime by running the Motion VI Library Help (MCLV.HLP) file.

Mclv.hlp


Motion VI Library Help
File Edit Bookmark Options Help
Contents Index Back Print << >>
I/O
Move
Set
Get
Motion VI Library Help
File Edit Bookmark Options Help
Contents Index Back Print << >>
Motion
Abort Dir Enab Go
Abs Rel Stop

Motion VI Library Version 2.0

Welcome to the online help for the Motion VI library. The Motion VI Library consists of over 60 separate VIs related to motion control. Version 2.0 of the Motion VI Library includes support for the Motion Dialog functions, high-level functions for configuration and initialization. This online help file contains detailed information about the VIs that make up the Library, and tips for working with LabVIEW.

Digital & Analog I/O

Motion Functions

Parameter Setup Functions

Reporting Functions

Motion VIs

The Motion Function Palette.

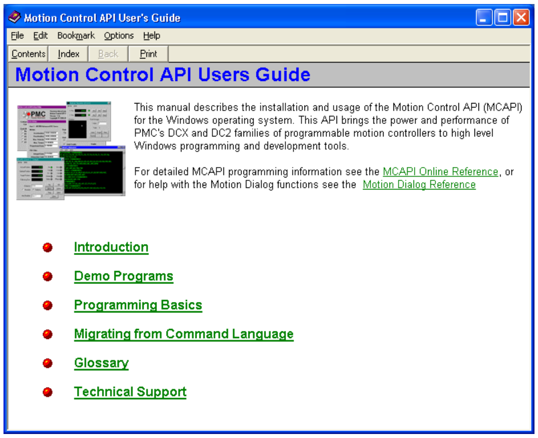
This section describes the VIs for the control of motion, including starting motion, stopping motion, and controlling the direction of travel.

MCAbort	MCMoveAbsolute
MCDirection	MCMoveRelative
MCEnableAxis	MCStop
MCGo	MCWait
MCGoHome	MCWaitForStop

The online Motion VI Library Reference provides detailed descriptions of available VI's.

You will also find online help for the Motion Control Application Programming Interface (MCAPI). The MCAPI contains all the possible functions that may be commanded of the board. We are continually adding Motion VIs to our Library, however, you may find functionality in the MCAPI that you would like to incorporated into your own VIs.

Mcguide.hlp


Motion Control API User's Guide
File Edit Bookmark Options Help
Contents Index Back Print

Motion Control API Users Guide

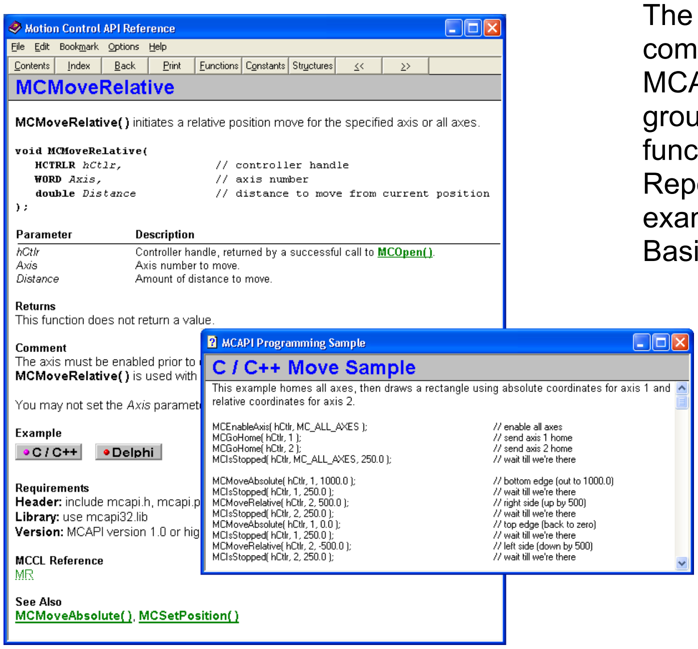
This manual describes the installation and usage of the Motion Control API (MCAPI) for the Windows operating system. This API brings the power and performance of PMC's DCX and DC2 families of programmable motion controllers to high level Windows programming and development tools.

For detailed MCAPI programming information see the MCAPI Online Reference, or for help with the Motion Dialog functions see the Motion Dialog Reference

- Introduction
- Demo Programs
- Programming Basics
- Migrating from Command Language
- Glossary
- Technical Support

The online MCAPI Users Guide describes the basics of PMC's MCAPI. This should be the '**first stop**' for any questions about the MCAPI.

Mcapi.hlp


Motion Control API Reference
File Edit Bookmark Options Help
Contents Index Back Print Functions Constants Structures << >>

MCMoveRelative

MCMoveRelative( ) initiates a relative position move for the specified axis or all axes.

```
void MCMoveRelative(
   HCTRLR hCtlr,               // controller handle
   WORD Axis,                  // axis number
   double Distance             // distance to move from current position
);
```

Parameter	Description
hCtlr	Controller handle, returned by a successful call to **MCOpen( )**.
Axis	Axis number to move.
Distance	Amount of distance to move.

Returns
This function does not return a value.

Comment
The axis must be enabled prior to
MCMoveRelative( ) is used with

You may not set the *Axis* paramete

Example

Requirements
Header: include mcapi.h, mcapi.p
Library: use mcapi32.lib
Version: MCAPI version 1.0 or hig

MCCL Reference
MR

See Also
MCMoveAbsolute( ), **MCSetPosition( )**

C / C++ Move Sample

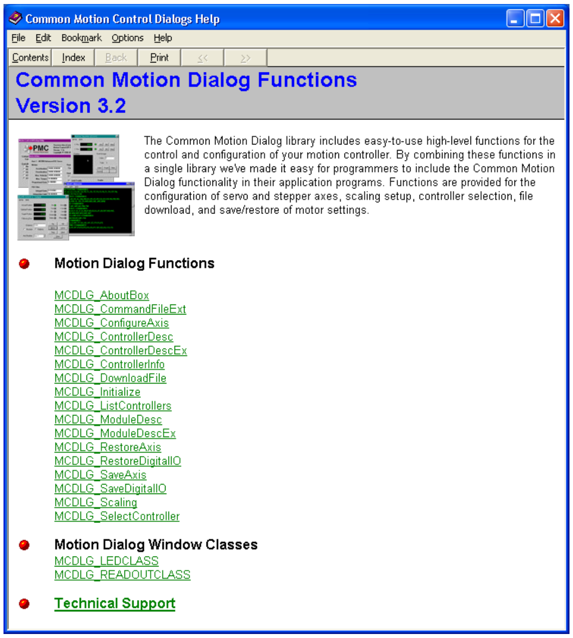
This example homes all axes, then draws a rectangle using absolute coordinates for axis 1 and relative coordinates for axis 2.

```
MCEnableAxis( hCtlr, MC_ALL_AXES );              // enable all axes
MCGoHome( hCtlr, 1 );                            // send axis 1 home
MCGoHome( hCtlr, 2 );                            // send axis 2 home
MCIsStopped( hCtlr, MC_ALL_AXES, 250.0 );        // wait till we're there

MCMoveAbsolute( hCtlr, 1, 1000.0 );              // bottom edge (out to 1000.0)
MCIsStopped( hCtlr, 1, 250.0 );                  // wait till we're there
MCMoveRelative( hCtlr, 2, 500.0 );               // right side (up by 500)
MCIsStopped( hCtlr, 2, 250.0 );                  // wait till we're there
MCMoveAbsolute( hCtlr, 1, 0.0 );                 // top edge (back to zero)
MCIsStopped( hCtlr, 1, 250.0 );                  // wait till we're there
MCMoveRelative( hCtlr, 2, -500.0 );              // left side (down by 500)
MCIsStopped( hCtlr, 2, 250.0 );                  // wait till we're there
```

The online MCAPI Reference provides a complete listing and description of all MCAPI functions. Function calls are grouped both alphabetically and by functional groups (Motion, Setup, Reporting, Gearing, etc...). Source code examples are provided for C++, Visual Basic, and Delphi.

Mcdlg.hlp



Common Motion Dialog Functions
Version 3.2

The Common Motion Dialog library includes easy-to-use high-level functions for the control and configuration of your motion controller. By combining these functions in a single library we've made it easy for programmers to include the Common Motion Dialog functionality in their application programs. Functions are provided for the configuration of servo and stepper axes, scaling setup, controller selection, file download, and save/restore of motor settings.

- **Motion Dialog Functions**

  MCDLG_AboutBox
  MCDLG_CommandFileExt
  MCDLG_ConfigureAxis
  MCDLG_ControllerDesc
  MCDLG_ControllerDescEx
  MCDLG_ControllerInfo
  MCDLG_DownloadFile
  MCDLG_Initialize
  MCDLG_ListControllers
  MCDLG_ModuleDesc
  MCDLG_ModuleDescEx
  MCDLG_RestoreAxis
  MCDLG_RestoreDigitalIO
  MCDLG_SaveAxis
  MCDLG_SaveDigitalIO
  MCDLG_Scaling
  MCDLG_SelectController

- **Motion Dialog Window Classes**
  MCDLG_LEDCLASS
  MCDLG_READOUTCLASS

- **Technical Support**

The online MCAPI Common Dialog Reference describes the high level MCAPI Dialog functions. These operations include: Save and Restore axis configurations (PID and Trajectory), Windows Class Position and Status displays, Scaling, and I/O configuration.

Chapter Contents

Chapter

2

Low-Level Communication

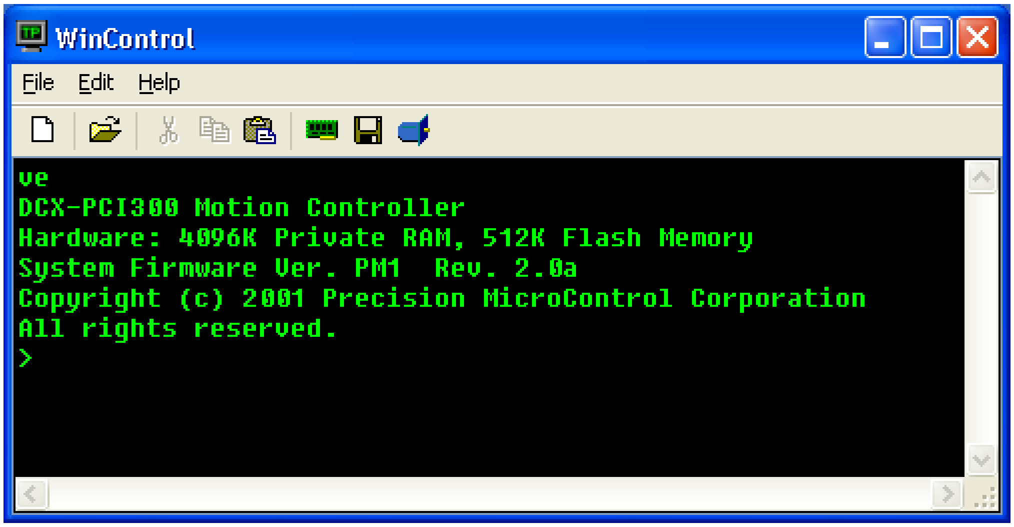
At its lowest level the operation of the motion control card is similar to that of a microprocessor, it has a predefined instruction set of operations which it can perform. This instruction set, known as **Motion Control Command Language** (MCCL), consists of over 200 operations which include motion, setup, conditional (if/then), mathematical, and I/O operations.

The typical PC based application will never call these low-level commands directly. Instead, the programmer will call higher-level language functions (in C++, Visual Basic, Delphi, or LabVIEW) which pass the appropriate native, board-level MCCL command(s) through the use of the MCAPI device driver. However, an understanding of how the low-level commands work allows better command of the Motion Library VIs, especially the Low-Level OEM VIs.

Win Control and MCCL Commands

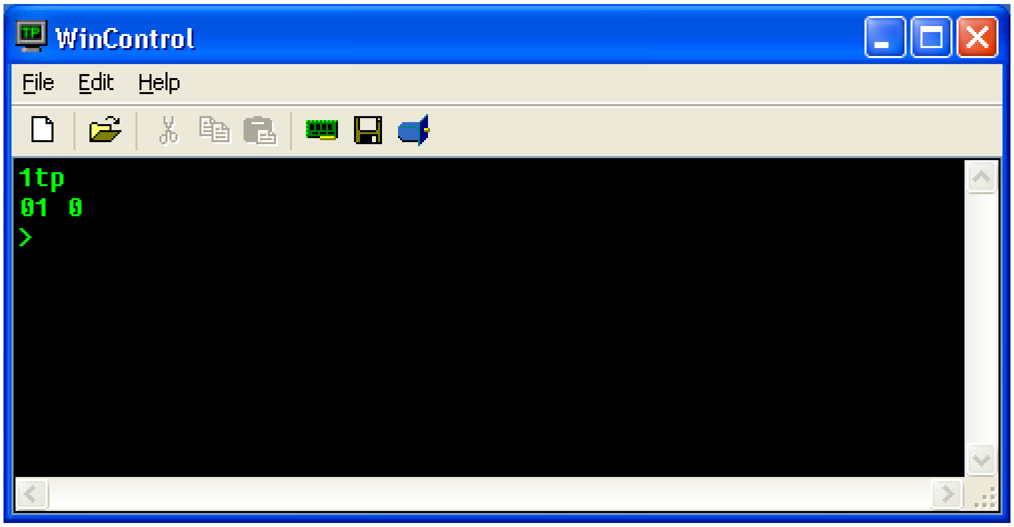
The Win Control utility allows the user to communicate with the motion control card in its native language (MCCL). This utility communicates with the controller via the PCI ASCII interface. All MCCL commands are described in detail in the **Motion Control Command Language (MCCL) Reference Manual** specific to your controller.

MCCL commands are two character alphanumeric mnemonics built with two key characters from the description of the operation (i.e.. "MR" for **M**ove **R**elative). When the command, followed by a carriage return, is received by the motion control card, it will be executed. The following graphic shows the result of executing the VE command. This command causes the motion control card to report firmware version and the amount of installed memory.



```
ve
DCX-PCI300 Motion Controller
Hardware: 4096K Private RAM, 512K Flash Memory
System Firmware Ver. PM1  Rev. 2.0a
Copyright (c) 2001 Precision MicroControl Corporation
All rights reserved.
>
```

All axis related MCCL commands will be preceded by an axis number, identifying to which axis the operation is intended. The following graphic shows the result of issuing the **T**ell **P**osition (aTP) command to axis number one.



```
1tp
01 0
>
```

Note that each character typed at the keyboard should be echoed to your display. If you enter an illegal character or an illegal series of valid characters, the motion control card will return a question mark character, followed by an error code. The **MCCL Error Code** listing can be found in the **Motion Control Command Language (MCCL) Reference Manual** specific to your controller. On receiving this response, you should re-enter the entire command/command string. If you make a mistake in typing, the backspace can be used to correct it. The motion control card will not begin to execute a command until a carriage return is received.

Once you are satisfied that the communication link is correctly conveying your commands and responses, you are ready to check the motor interface. When the motion control card is powered up or reset, each motor control module is automatically set to the "motor off" state. In this state, there should be no drive current to the motors. For servos it is possible for a small offset voltage to be present. This is usually too small to cause any motion, but some systems have so little friction or such high amplifier gain, that a few millivolts can cause them to drift in an objectionable manner. If this is the case, the "null" voltage can be minimized by adjusting the offset adjustment potentiometer on the respective servo control module.

Before a motor can be successfully commanded to move certain parameters must be set by issuing commands to the motion control card. These include; PID filter gains, trajectory parameters (maximum velocity, acceleration, and deceleration), allowable following error, configuring motion limits (hard and soft).

At this point the user should refer to the Motion Control chapter and the sections that deal with Theory of Motion Control, Servo Basics and Stepper Basics in the appropriate **User's Manual** for the motion control card you are using. There the you will find more specific information for each type of motor, including which parameters must be set before a motor should be turned on and how to check the status of the axis.

Assuming that all of the required motor parameters have been defined, the axis is enabled with the **M**otor o**N** (aMN) command. Parameter 'a' of the **M**otor o**N** command allows the user to turn on a specific axis or all axes. To enable all, enter the **M**otor o**N** command with parameter 'a' = 0. To enable a single axis issue the **M**otor o**N** command where 'a' = the axis number to be enabled.

After turning a particular axis on, it should hold steady at one position without moving. The **T**ell **T**arget (aTT) and **T**ell **P**osition (aTP) commands should report the same number. There are several commands which are used to begin motion, including **M**ove **A**bsolute (MA) and **M**ove **R**elative (MR). To move axis 2 by 1000 encoder counts, enter 2MR1000 and a carriage return. If the axis is in the "**M**otor o**N**" state, it should move in the direction defined as positive for that axis. To move back to the previous position enter 2MR-1000 and a carriage return.

With the any of PMC's motion controllers, it is possible to group together several commands. This is not only useful for defining a complex motion which can be repeated by a single keystroke, but is also useful for synchronizing multiple motions. To group commands together, simply place a comma between each command, pressing the return key only after the last command.

A repeat cycle can be set up with the following compound command:

```
2MR1000,WS0.5,MR-1000,WS0.5,RP6  <return>
```

This command string will cause axis 2 to move from position 1000 to position –1000 7 times. The **R**e**P**eat (RP) command at the end causes the previous command to be repeated 6 additional times. The **W**ait for **S**top (WS) commands are required so that the motion will be completed (trajectory complete) before the return motion is started. The number 0.5 following the WS command specifies the number of seconds to wait after the axis has ceased motion to allow some time for the mechanical components to come to rest and reduce the stresses on them that could occur if the motion were reversed instantaneously. Notice that the axis number need be specified only once on a given command line.

A more complex cycle could be set up involving multiple axes. In this case, the axis that a command acts on is assumed to be the last one specified in the command string. Whenever a new command string is entered, the axis is assumed to be 0 (all) until one is specified.

Entering the following command:

```
2MR1000,3MR-500,0WS0.3,2MR1000,3MR500,0WS0.3,RP4     <return>
```

will cause axis 2 to move in the positive direction and axis 3 to move in the negative direction. When both axes have stopped moving, the WS command will cause a 0.3 second delay after which the remainder of the command line will be executed.

After going through this complex motion 5 times, it can be repeated another 5 times by simply entering a return character. All command strings are retained by the controller until some character other than a return is entered. This comes in handy for observing the position display during a move. If you enter:

```
1MR1000      <return>
1TP          <return>
(return)
(return)
(return)
(return)
```

The motion control card will respond with a succession of numbers indicating the position of the axis at that time. Many terminals have an "auto-repeat" feature which allows you to track the position of the axis by simply holding down the return key.

Another way to monitor the progress of a movement is to use the **R**e**P**eat command without a value. If you enter:

```
1MR10000     <return>
1TP,RP       <return>
```

The position will be displayed continuously. These position reports will continue until stopped by the operator pressing the Escape key.

While the motion control card is executing commands, it will ignore all alphanumeric keys that are pressed. The user can abort a currently executing command or string by pressing the escape key. If the user wishes only to pause the execution of commands, the user should press the space bar. In order to restart command execution press the space bar again. If after pausing command execution, the user decides to abort execution, this can be done by pressing the escape key.

Chapter Contents

- Samples
- The Execute Input
- Cascading VIs
- Self-Documenting Constants

Chapter

3

Understanding LabVIEW

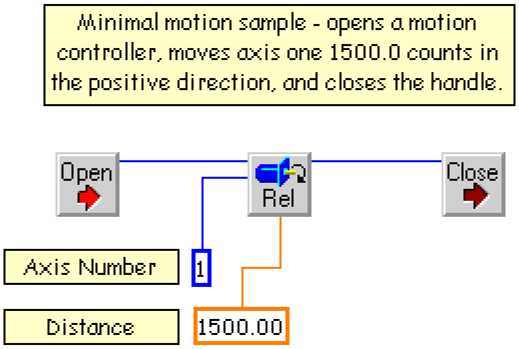
Obviously we cannot hope to teach you LabVIEW in a single chapter. Instead, you will find information to supplement what you already know. We provide samples as part of the Motion VI Library installation that will give you working code from which you may build upon. We would also like to show you that the Motion VI Library was built with commonality in mind. The execute VI and the cascading of VIs sections will show you how to streamline your code for performance and clarity. The self-documenting constants section may be review, however, the example shows that we do support such prudent programming practices.

Samples

Four sample programs are now included with the Motion VI library. The first, **SIMPLE.VI**, shows how to execute a simple move. The **SAMPLE.VI** sample provides an interactive panel for moving an axis and monitoring the status of that axis. **CYCLE.VI** demonstrates how to implement a state machine and execute multiple moves under program control (the state machine approach makes it easy to monitor the status of axes while the motions are executed). Finally, **ANALOG.VI** demonstrates the use of the auxiliary analog inputs available on most PMC motion controllers.

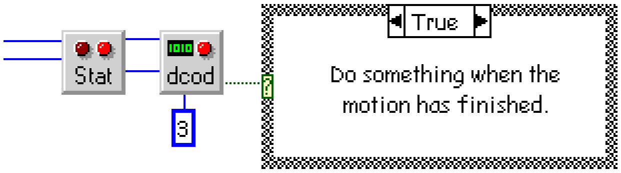
The Motion VIs are installed in the Instrument Drivers function palette in a number of logically arranged sub-palettes. To better see how the VIs are used, open the **SAMPLE.VI** from the file menu (select File | Open, select the INSTR.LIB directory, then the MOTION CONTROL directory, and finally **SAMPLE.VI**).

The first step in any motion program is to obtain a handle to the controller, using the **MCOpen** VI. This handle is used in all subsequent calls to the Motion VIs. When the program completes the handle should be passed to the **MCClose** VI to ensure the motion controller is properly closed. Failure to properly close the handle is the primary source of errors when using the Motion VI Library. The following wiring diagram, from the **SIMPLE.VI** sample program, demonstrates how to open the motion controller, perform a simple move, and close the motion controller:



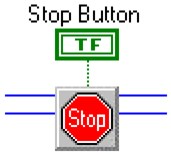
Minimal motion sample - opens a motion controller, moves axis one 1500.0 counts in the positive direction, and closes the handle.

A common question is how best to wait for a motion to complete. The preferred method is to use **MCGetStatus.vi** and **MCDecodeStatus.vi** to test each axis involved in the motion for trajectory complete. By placing the testing in a loop you are able to perform other processing while waiting for the motion to complete (such as updating front panel displays). The **CYCLE.VI** sample demonstrates this technique.

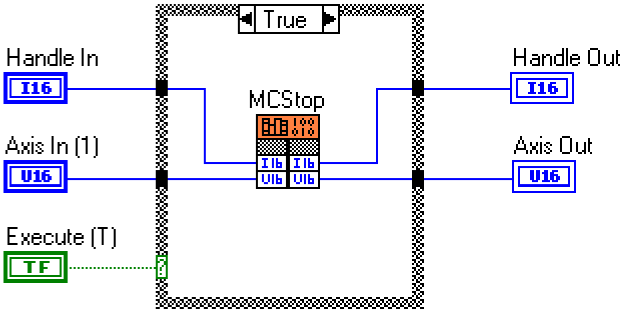


Do something when the motion has finished.

The Execute Input



MCStop VI with Execute Input Wired to Control

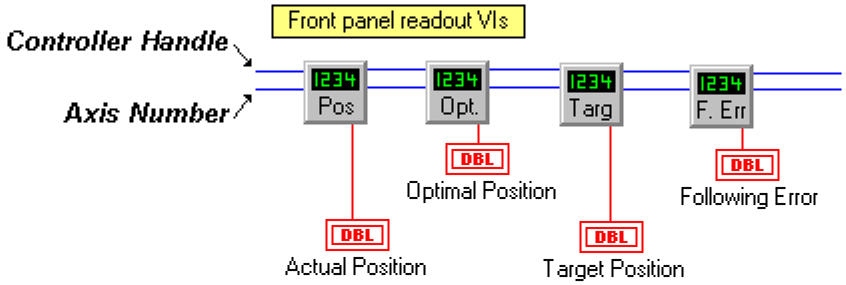


MCStop VI Diagram

When working with a complex motion control card, there are often times when a particular VI only needs to execute in response to a change of settings, such as the user pressing the STOP button. This is often accomplished by surrounding the VI with control logic. The Motion Control VIs have this logic built in! Default behavior is to execute immediately, but if the user chooses a Boolean control may be wired to the **Execute** input to control execution of the VI.

The Sample VI included with the Motion VI Library demonstrates the use of the Execute input, where the On, Off, Stop, Abort, Home, and Zero front panel controls are connected directly to Execute inputs on their respective Motion VI Library VIs.

Cascading VIs



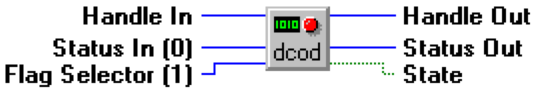
In many cases, you will find it necessary to wire together several of the Motion Control VIs in order to achieve a particular level of control. To simplify wiring, the Motion Control VIs support cascading, where common inputs (the controller handle and axis number) are echoed back out of the VI and may be used to provide those some signals to the next VI in the chain.

One useful side effect of this design is that it may be used to control order of execution. Since LabVIEW will not start a VI executing until all of its inputs are available cascaded VIs will execute in order. If the same VIs were wired in parallel it would not be possible to determine the order of execution.

The **SAMPLE.VI** included with the Motion VI Library demonstrates the use of cascaded VIs for most of the controls and displays on the front panel.

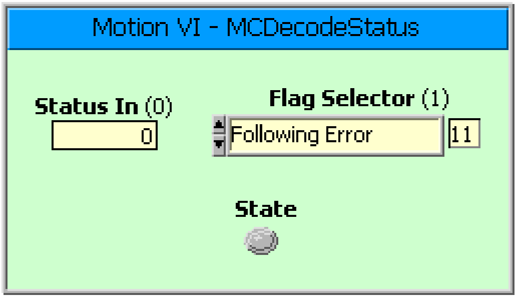
Self-Documenting Constants

Many of the VIs have one input that will take several different constants to yield different output. For instance, the **MCDecodeStatus** VI's Flag Selector will take different values, and turn on the corresponding LED on the front panel depending on the state of that flag chosen by Flag Selector.



MCDecodeStatus.vi

By double clicking on the **MCDecodeStatus** VI you will see the following panel appear on your screen.

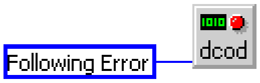

Motion VI - MCDecodeStatus

Status In (0): 0

Flag Selector (1): Following Error 11

State

You may choose the status you are interested in by clicking on the Flag Selector box with the LabVIEW hand tool. In this case, we would like to monitor whether or not we have exceeded the preset following error. By using the LabVIEW arrow tool, you may drag and drop the Flag Selector box into the wiring diagram where you may then wire the value to the **MCDecodeStatus** VI.



Instead of hard coding in the value of 11, you will now have descriptive text that will be much more meaningful in the unfortunate event that someone would need to debug your code. Just remember, that unfortunate someone may be you several months after it was written.

Chapter Contents

Chapter 4

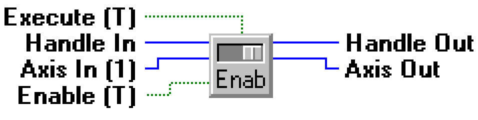
Motion VI Library Introduction

This brief chapter gives an example of a VI listing and will hopefully familiarize you with what information each of the sections gives. Not all sections will be listed under each VI, but following the example are each of the section headings that can be found. A description follows the section heading informing what information you may be likely to find.

VI Listing Introduction

An example of a VI listing is shown below. What follows the example is a brief description of what should be found in each of the respective headings.

MCEnableAxis



MCEnableAxis.vi

MCEnableAxis turns the specified axis on if **Enable** is TRUE, or off if **Enable** is FALSE. Note that an axis must be enabled before any motion will take place.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to enable/disable.

TF **Enable** enables the selected axis if it is TRUE, or disables the axis if it is FALSE.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

This VI does much more than just enable or disable **Axis In**. However, as the name implies, the selected axis(axes) will be turned on or off depending upon the value of **Enable**. Note that an axis must be enabled before any motion will take place. Issuing this command with **Axis In** set to ALL AXES (a value of zero) will enable or disable all axes installed on **Handle In**.

If **Axis In** is off and then turned on, the following events will occur.

- The target and optimal positions are set to the present encoder position.
- The data passed by **MCSetScale** are applied.
- MC_STAT_AMP_ENABLE will be set.
- MC_STAT_AMP_FAULT, if present, will be cleared.
- MC_STAT_ERROR, if present, will be cleared.
- MC_STAT_FOLLOWING, if present, will be cleared.
- MC_STAT_MLIM_TRIP, if present, will be cleared.
- MC_STAT_MSOFT_TRIP, if present, will be cleared.
- MC_STAT_PLIM_TRIP, if present, will be cleared.
- MC_STAT_PSOFT_TRIP, if present, will be cleared.

If **Axis In** is on and then turned on again, the following events will occur.

- The data passed by **MCSetScale** are applied.

> ⚠ Calling this function to enable or disable an axis while it is in motion is not recommended. However, should it be done, **Axis In** will cease the current motion profile, and MC_STAT_AT_TARGET will be set.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

MCCL Reference

MF, MN

See Also

MCAbort, **MCStop**

Each VI listing begins with a picture of the VI and a brief introductory description that explains for what the VI is used.

Parameters then further explains in detail the purpose of each parameter. If one of the parameters is a cluster, a following table holds all of the members and a brief description for each member.

Comments describes the VI in more detail. Explanation will range from why the VI is used, to how it is used, where it could cause problems and potential alternatives.

Occasionally, the following two boxes can be found in the comments section and contain relevant information that needs to be emphasized. The first box aids in the understanding of the function. The second box warns of scenarios that will more than likely cause problems.

> ℹ Information to assist the programmer.

> ⚠ Warning to help the programmer avoid potential problems.

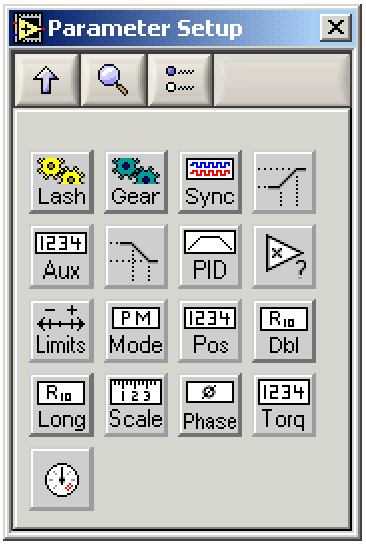
Compatibility gives information as to which motion control cards or modules will not work with the function. Generally, only exceptions will be listed, as to provide a more concise listing.

Requirements lists the earliest version of the MCAPI and the Motion VI Library that are necessary to use this VI.

MCCL Reference lists the MCCL level commands that comprise the high level function. More information can be found in the **Motion Control Command Language (MCCL) Reference Manual** specific to your controller on how each of these commands works. Not all functions will be comprised of speaking to the board with MCCL commands, in which cases there will be no equivalent commands.

See Also lists related VIs. Some of these VIs may be alternatives to be used, while others may be the corresponding get VI to a set VI. Yet there will be other VIs that must be used as in tandem with another VI.

Chapter Contents



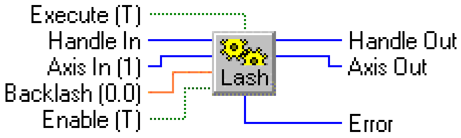
Chapter

5

Parameter Setup VIs

Parameter setup VIs allow the program to consistently configure the motion control card and individual modules to behave in an appropriate manner for a given application. Although trajectory parameters, PID loop gains, and end of travel limits should be set prior to commanding motion, these and other parameters may be changed during a move. However, certain parameters once passed to the card will not alter behavior until **MCEnableAxis** is called, which allows the specific axis to then implement several queued parameters at once in a logical and safe fashion. For first time setup, a development tool like **Motion Integrator** should be used to determine the proper tuning parameters that can be passed by the functions in this chapter.

MCEnableBacklash



MCEnableBacklash.vi

The **MCEnableBacklash** VI sets the backlash compensation distance and turns backlash compensation on or off, depending upon the value of **Enable**.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to enable/disable the backlash of.

DBL **Backlash** is the amount of backlash compensation to apply.

TF **Enable** set to TRUE to enable backlash compensation, FALSE to disable.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

In applications where the mechanical system is not directly connected to the motor, it may be required that the motor move an extra amount to take up gear backlash. The **Backlash** parameter to this VI sets the amount of this compensation, and should be equal to one half of the amount the axis must move to take up the backlash when it changes direction.

Compatibility

Stepper axes, the DC2, DCX-PC, and DCX-PCI100 controllers do not support backlash compensation.

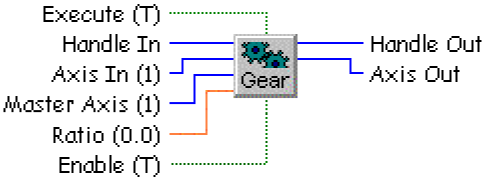
Requirements

MCAPI: version 2.0 or higher
Motion VI Library: version 2.0 or higher

MCCL Reference

BD, BF, BN

MCEnableGearing



MCEnableGearing.vi

The **MCEnableGearing** VI enables or disables electronic gearing for the specified **Axis In** / **Master Axis** pair.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to enable/disable gearing for.

U16 **Master Axis** selects the controlling axis (i.e. master) for this axis.

DBL **Ratio** the gearing ratio between this axis and the master.

TF **Enable** set to TRUE to enable gearing, FALSE to disable.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

This function permits you to configure one axis to automatically reproduce the motions of a master axis. In addition, by using a ratio of other than 1.0, the reproduced motion can be scaled as desired.

DC2 users should express the ratio as a floating point value (i.e. 0.5 for 2:1, 2.0 for 1:2, etc.). **MCEnableGearing** automatically converts this ratio to the 32 bit fixed point fraction the DC2 requires. The DCX-PC100 controller supports only a fixed ration of 1:1, the Ratio parameter is ignored for this controller.

Compatibility

The DCX-PCI100 controller, DC2 stepper axes, the MC150, MC160, MC200, and MC260 modules when placed on the DCX-PC100 controller do not support gearing.

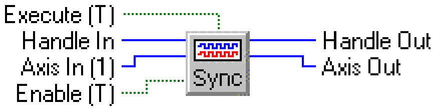
Requirements

MCAPI: version 1.0 or higher
Motion VI Library: Version 2.0 or higher

MCCL Reference

SM, SS

MCEnableSync



MCEnableSync.vi

The **MCEnableSync** VI enables or disables synchronized motion for contour path motion for the specified axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to enable/disable synchronized motion for.

TF **Enable** set to TRUE to enable synchronized motion, FALSE to disable.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

This function is issued to the controlling axis of a contour path motion, prior to issuing any contour path motions, to inhibit any motion until a call to **MCGo** is made.

Compatibility

The MCAPI does not does not support contouring on the DC2, DCX-PC100, or DCX-PCI100 controllers.

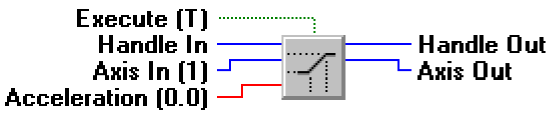
Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 2.0 or higher

MCCL Reference

NS, SN

MCSetAcceleration



MCSetAcceleration.vi

The **MCSetAcceleration** VI sets the programmed acceleration value for the selected axis to **Acceleration**, where **Acceleration** is specified in the current units for the axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the acceleration of.

DBL **Acceleration** is the new acceleration value for axis.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

A value of zero may be specified for **Axis In** to set the acceleration for all axes installed on a controller.

Compatibility

The DC2 stepper axes do not support ramping.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

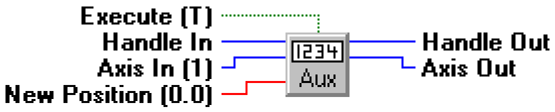
MCCL Reference

SA

See Also

MCGetAccelerationEx

MCSetAuxEncPos



MCSetAuxEncPos.vi

MCSetAuxEncPos sets the current position of the auxiliary encoder.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the auxiliary encoder position of.

DBL **Position** is the new auxiliary encoder position value for axis.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

This VI sets the current position of the auxiliary encoder to the value given by the **Position** parameter. A value of zero may be specified for **Axis In** to set the auxiliary encoders for all axes installed on a controller.

> DCX-AT200 firmware version 3.5a or higher, or DCX-PC100 firmware version 4.9a or higher is required if you wish to set the position of the auxiliary encoder to a value other than zero. Earlier firmware versions ignore the value in the Position argument and zero the Auxiliary Encoder.

Compatibility

The DC2, DCX-PCI100 controllers, MC100, MC110, MC150, and MC320 modules do not support auxiliary encoders. Closed-loop steppers do not support auxiliary encoder functions, since the connected encoder is considered a primary encoder.

Requirements

MCAPI: version 1.0 or higher

Motion VI Library: version 1.1 or higher

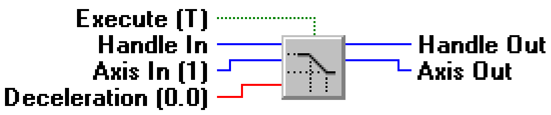
MCCL Reference

AH

See Also

MCGetAuxEncPosEx

MCSetDeceleration



MCSetDeceleration.vi

The **MCSetDeceleration** VI sets the programmed deceleration value for **Axis In** to **Deceleration**, where **Deceleration** is specified in the current units for the axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the deceleration of.

DBL **Deceleration** is the new deceleration value for axis.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

A value of ALL AXES (0) may be specified for **Axis In** to set the deceleration for all axes installed on a controller.

Compatibility

The DCX-PCI100 controller, MC100, MC110, MC150, and MC160 modules do not support a separate deceleration value. Instead, the acceleration value will also be used as the deceleration value. The DC2 stepper axes do not support ramping.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

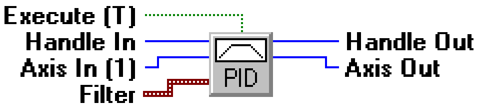
MCCL Reference

DS

See Also

MCGetDecelerationEx

MCSetFilterConfig



MCSetFilterConfig.vi

MCSetFilterConfig sets the PID loop for axis to the configuration given by the **Filter** cluster.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the PID filter of.

Filter is a cluster containing the new PID filter values for axis. The **Filter** cluster contains the following values:

DBL	**Derivative Gain** sets the derivative term of the PID loop.
DBL	**DerSample Period** is the time interval, in seconds, between derivative samples.
DBL	**Integral Gain** sets the integral term of the PID loop.
DBL	**Integration Limit** limits the power the integral gain can use to reduce error to zero.
DBL	**Velocity Gain** sets the feed-forward gain of the PID loop, volts per encoder count per second.
DBL	**Acceleration Gain** sets the feed-forward acceleration gain setting.
DBL	**Deceleration Gain** sets the feed-forward deceleration gain setting.
DBL	**Following Error** is the maximum position error, default units are encoder counts.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

The easiest way to change filter settings is to first call **MCGetFilterConfig** to obtain the current PID filter settings for **Axis In**, modify the values in the **Filter** cluster, and write the changed settings back to **Axis In** with **MCSetFilterConfig**.

> Closed-loop stepper operation requires firmware version 2.1a or higher on the DCX-PCI300 and firmware version 2.5a or higher on the DCX-AT300.

Compatibility

Velocity Gain is not supported on the DCX-PCI100 controller, MC100, MC110 modules, or closed-loop steppers. Acceleration Gain is not supported on the DC2, DCX-PC100, or DCX-PCI100 controllers. Deceleration Gain is not supported on the DC2, DCX-PC100, or DCX-PCI100 controllers.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

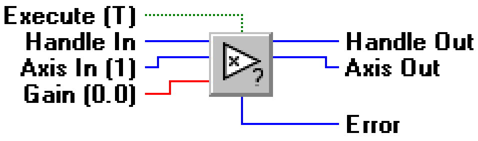
MCCL Reference

AG, DG, FR, IL, SD, SE, SI, VG

See Also

MCGetFilterConfig

MCSetGain



MCSetGain.vi

The **MCSetGain** VI sets the proportional gain of a servo's feedback loop.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the gain of.

DBL **Gain** is the new gain value for axis.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

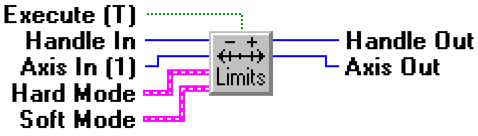
MCCL Reference

SG

See Also

MCGetGain

MCSetLimits



MCSetLimits.vi

MCSetLimits sets the hard and soft limits for the selected axis. Motion controllers that do not support soft limits ignore the soft limit settings.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the limits of.

▤ **Hard Mode** is a cluster containing the new hard limit settings for axis. The cluster values are as follows:

TF	**Low Limit** set to TRUE enables the lower hard limit.
TF	**High Limit** set to TRUE enables the upper hard limit.
U16	**Mode** selects stop mode – Turn Off (0) / Abrupt (4) / Smooth (8)

▤ **Soft Mode** is a cluster containing the new soft limit settings for axis. The cluster values are as follows:

TF	**Low Limit** set to TRUE enables the lower soft limit.
TF	**High Limit** set to TRUE enables the upper soft limit.
U16	**Mode** selects stop mode – Turn Off (0) / Abrupt (4) / Smooth (8)
DBL	**Low Set** sets the lower soft limit value.
DBL	**High Set** sets the upper soft limit value.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Compatibility

The DC2 and DCX-PC100 controllers do not support soft limits.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.1 or higher

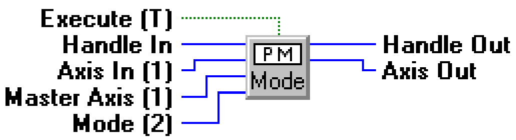
MCCL Reference

HL, LF, LL, LM, LN

See Also

MCGetLimits

MCSetOperatingMode



MCSetOperatingMode.vi

The **MCSetOperatingMode** VI sets the controller operating mode for **Axis In***.*

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the operating mode of.

U16 **Master Axis** selects the master axis for master/slave mode.

U16 **Mode** is the new operating mode for axis. Set to 0 for contour mode, 1 for gain mode, 2 for position mode (the default), 3 for torque mode, or 4 for velocity mode.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

This VI is used to switch between the main operating modes of the controller. All modes except contouring are supported by all controllers.

⚠ This VI should not be called while **Axis In** is in motion.

Compatibility

The MCAPI does not does not support contouring on the DC2, DCX-PC100, or DCX-PCI100 controllers. Gain mode is not supported on stepper axes, MC100, or MC110 modules. Torque mode is not supported on stepper axes, DCX-PCI100 controller, MC100, or MC110 modules.

Requirements

MCAPI: version 1.0 or higher

Motion VI Library: version 1.0 or higher

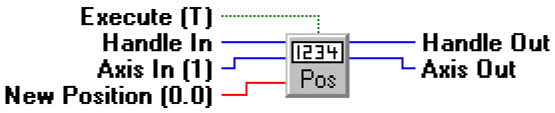
MCCL Reference

CM, GM, PM, QM, VM

See Also

Controller hardware manual

MCSetPosition



MCSetPosition.vi

The **MCSetPosition** VI sets the current position for the axis to **Position**.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the position of.

DBL **Position** is the new position value for axis.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

The current position of **Axis In** will be immediately updated to the value of **Position**.

This function may be called with **Axis In** set to ALL AXES (a value of zero) to set the position of all axes at once. All axes will be set to the same value of **Position**.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

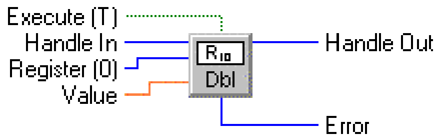
MCCL Reference

DH

See Also

MCGetPositionEx

MCSetRegisterDouble



MCSetRegisterDouble.vi

The **MCSetRegisterDouble** VI sets the value of a motion controller register to the specified double precision floating point value.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Register** selects the register to set the value of.

DBL **Value** is the new value for the register.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

When running background tasks on a multitasking controller the only way to communicate with the background tasks is to pass parameters in the general purpose registers. You cannot write to the local registers (registers 0 - 9) of a background task. When you need to communicate with a background task be sure to use one or more of the global registers (10 - 255).

Requirements

MCAPI: version 2.0 or higher
Motion VI Library: version 2.0 or higher

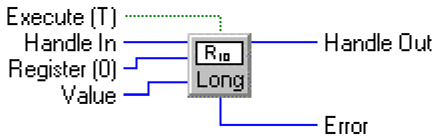
MCCL Reference

AL, AR

See Also

MCGetRegisterDouble, **MCGetRegisterLong**, **MCSetRegisterLong**

MCSetRegisterLong



MCSetRegisterLong.vi

The **MCSetRegisterLong** VI sets the value of a motion controller register to the specified 32-bit integer value.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Register** selects the register to set the value of.

I32 **Value** is the new value for the register.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

When running background tasks on a multitasking controller the only way to communicate with the background tasks is to pass parameters in the general purpose registers. You cannot write to the local registers (registers 0 - 9) of a background task. When you need to communicate with a background task be sure to use one or more of the global registers (10 - 255).

Requirements

MCAPI: version 2.0 or higher
Motion VI Library: version 2.0 or higher

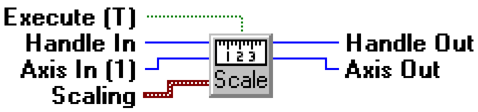
MCCL Reference

AL, AR

See Also

MCGetRegisterDouble, **MCGetRegisterLong**, **MCSetRegisterDouble**

MCSetScale



MCSetScale.vi

The **MCSetScale** VI sets scaling for the specified axis to the values contained in the **Scaling** cluster.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to abort the motion of.

Scaling is a cluster containing the new scale factors for axis.

DBL	**Constant** acts as a scale factor for servo analog outputs. By calibrating your motor/amplifier combination, it is possible to scale the output with **Constant** so that torque settings may be specified directly in ft-lbs.
DBL	**Offset** represents an offset from a servo encoder's index pulse to a zero position.
DBL	**Rate** acts as a multiplier for motion commands time values. The base controller time unit is the second, to convert this to minutes set **Rate** to 60.0, to convert to milliseconds rate should be set to 0.001
DBL	**Scale** is applied to motion parameters to convert from encoder counts to real world units.
DBL	**Zero** specifies that a soft zero should be located this distance from actual zero. By moving the soft zero around it is possible to have a series of position commands repeated at various spots in the range of travel without modifying the position commands. The actual zero position is not changed by this command.
DBL	**Time** is the time factor for controller level wait commands. See the discussion of the **Rate** parameter above for more information on setting this value. Note that a single **Time** value is maintained per controller (i.e. **Time** is axis independent).

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

Setting scaling factors allows application programs to talk to the controller in real world units, as opposed to arbitrary "encoder counts".

This function may be called with **Axis In** set to ALL AXES (a value of zero) to set the scaling of all axes at once. All axes will be set to the same value.

> ⚠ When you set **Scale** of the **Scaling** cluster to a value other than one, **Low Set** and **High Set** of the **Soft Mode** cluster should be changed to accommodate the new "real world" units.

Compatibility

The DC2 and the DCX-PC100 do not support any scaling members. The DCX-PCI100 does not support **Offset** or **Constant**.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

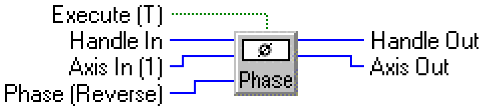
MCCL Reference

UK, UO, UR, US, UT, UZ

See Also

MCGetScale

MCSetServoOutputPhase



MCSetServoOutputPhase.vi

The **MCSetServoOutputPhase** VI sets the output phasing for the specified axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the phase of.

I16 **Phase** selects the phasing mode for this axis. On power up all axes have their phasing set to STANDARD (**Phase = 1**). Setting **Phase = 2** (the default for this VI) will select REVERSE phasing.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

This function may be called with **Axis In** set to ALL AXES (a value of zero) to set the phase of all axes at once. All axes will be set to the same value of **Phase**.

Compatibility

The MC100 and MC110 modules do not support phase reverse.

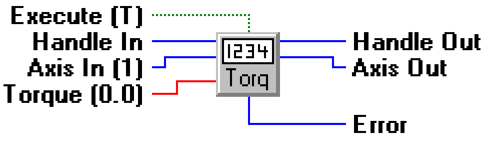
Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 2.0 or higher

MCCL Reference

PH

MCSetTorque



MCSetTorque.vi

The **MCSetTorque** VI sets maximum torque level for servos.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the torque of.

DBL **Torque** is the new torque value for axis.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Compatibility

Torque mode is not supported on stepper axes, DCX-PCI100 controller, MC100, or MC110 modules.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

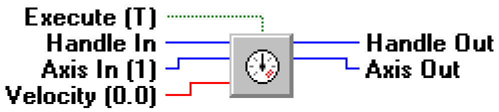
MCCL Reference

SQ

See Also

MCGetTorque

MCSetVelocity



MCSetVelocity.vi

The **MCSetVelocity** VI sets programmed velocity for the selected axis to **Velocity**, where **Velocity** is specified in the current units for axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the velocity of.

DBL **Velocity** is the new velocity value for axis.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

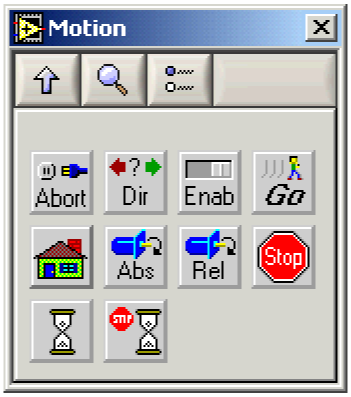
MCCL Reference

SV

See Also

MCGetVelocityEx

Chapter Contents



Chapter

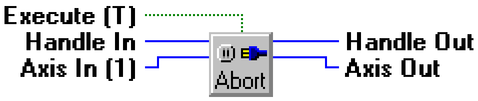
6

Motion VIs

Motion VIs range in use from allowing the program to commence or cease motion to permitting control of sequencing to altering operation of axes during motion.

A word of caution must be given regarding the use of board-level sequencing commands. Even though each of these functions includes a warning in this chapter, it should be stressed that once a command containing the word “Wait” or “Find” in the VI name is executed, no other VIs will be able to communicate until the board has completed what it was initially told to do. This can lead to scenarios where the calling program has absolutely no control during potentially dangerous or otherwise expensive situations.

MCAbort



MCAbort.vi

MCAbort aborts the current motion for the axis specified.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to abort the motion of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

The selected **Axis In** will execute an emergency stop following this command. Issuing this command with **Axis In** set to ALL AXES (a value of zero) will abort motion for all axes installed on the motion controller.

Servo axes will stop abruptly, and the servo control loop will remain energized.

For stepper motors, pulses from the motion controller will be disabled immediately. The state of the axis (enabled or disabled) following the call to **MCAbort** will depend upon the type of controller (see your controller hardware manual).

> i Following a call to **MCAbort**, verify that the axis has stopped using **MCWaitForStop**. Then call **MCEnableAxis** prior to issuing another motion command.

> i Following a call to **MCAbort** on the DCX-PC100 controller when in velocity mode, call **MCSetOperatingMode** prior to issuing another motion command.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

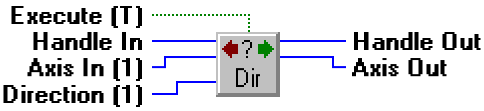
MCCL Reference

AB

See Also

MCEnableAxis, **MCSetOperatingMode**, **MCStop**, **MCWaitForStop**

MCDirection



MCDirection.vi

The **MCDirection** VI sets the direction of motion when operating in velocity mode.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the direction of.

U16 **Direction** selects the direction of travel for the axis. Set this parameter to 1 for FORWARD, or 2 for REVERSE.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

This command may be used to change the direction of travel when an axis is operating in Velocity Mode. The actual direction of travel for FORWARD and REVERSE and will depend upon your hardware configuration.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

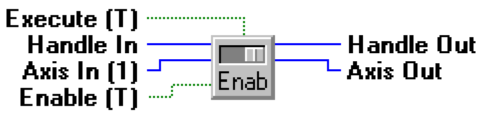
MCCL Reference

DI

See Also

MCSetOperatingMode

MCEnableAxis



MCEnableAxis.vi

MCEnableAxis turns the specified axis on if **Enable** is TRUE, or off if **Enable** is FALSE. Note that an axis must be enabled before any motion will take place.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to enable/disable.

TF **Enable** enables the selected axis if it is TRUE, or disables the axis if it is FALSE.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

This VI does much more than just enable or disable **Axis In**. However, as the name implies, the selected axis(axes) will be turned on or off depending upon the value of **Enable**. Note that an axis must be enabled before any motion will take place. Issuing this command with **Axis In** set to ALL AXES (a value of zero) will enable or disable all axes installed on **Handle In**.

If **Axis In** is off and then turned on, the following events will occur.

- The target and optimal positions are set to the present encoder position.
- The data passed by **MCSetScale** are applied.
- MC_STAT_AMP_ENABLE will be set.
- MC_STAT_AMP_FAULT, if present, will be cleared.
- MC_STAT_ERROR, if present, will be cleared.
- MC_STAT_FOLLOWING, if present, will be cleared.
- MC_STAT_MLIM_TRIP, if present, will be cleared.
- MC_STAT_MSOFT_TRIP, if present, will be cleared.
- MC_STAT_PLIM_TRIP, if present, will be cleared.
- MC_STAT_PSOFT_TRIP, if present, will be cleared.

If **Axis In** is on and then turned on again, the following events will occur.

- The data passed by **MCSetScale** are applied.

> ⚠ Calling this function to enable or disable an axis while it is in motion is not recommended. However, should it be done, **Axis In** will cease the current motion profile, and MC_STAT_AT_TARGET will be set.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

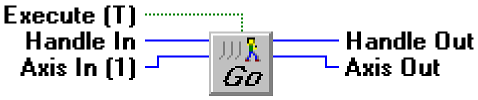
MCCL Reference

MF, MN

See Also

MCAbort, **MCStop**

MCGo



MCGo.vi

MCGo initiates a motion when operating in velocity mode. The axis must be configured for velocity mode operation before using **MCGo**.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to trigger.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

The axis must be configured for velocity mode operation before issuing a **MCGo** call. All axes may be instructed to move by setting the **Axis In** parameter to ALL AXES (a value of zero).

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

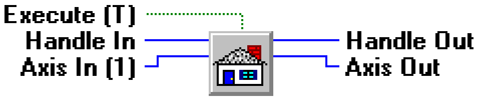
MCCL Reference

GO

See Also

MCSetOperatingMode, **MCStop**

MCGoHome



MCGoHome.vi

MCGoHome initiates a home motion for the specified axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to home.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

The home or zero position is relative to the position that was set using the **MCSetPosition** VI. This VI effectively executes an **MCMoveAbsolute** with a target position of 0.0.

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

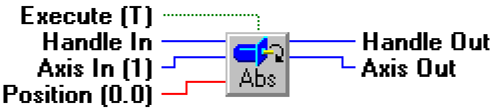
MCCL Reference

GH

See Also

MCMoveAbsolute, **MCSetPosition**

MCMoveAbsolute



MCMoveAbsolute.vi

MCMoveAbsolute initiates an absolute position move for the specified axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to move.

DBL **Position** is the new absolute position for the axis to move to.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

The axis must be enabled prior to executing a move.

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

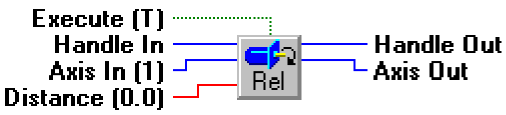
MCCL Reference

MA

See Also

MCMoveRelative, **MCSetPosition**

MCMoveRelative



MCMoveRelative.vi

MCMoveRelative initiates a relative position move for the specified axis. The axis must be enabled prior to executing a move.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to move.

DBL **Distance** is the relative distance to for the axis to move to.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

The axis must be enabled prior to executing a move.

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

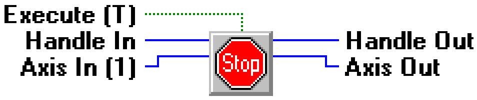
MCCL Reference

MR

See Also

MCMoveAbsolute, **MCSetPosition**

MCStop



MCStop.vi

The **MCStop** VI stops the specified axis using the pre-programmed deceleration values.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to stop the motion of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

This function initiates a controlled axis stop, as compared with **MCAbort** which stops the axis abruptly.

Following a call to **MCStop** verify that the axis has stopped using **MCWaitForStop**. Then call **MCEnableAxis** prior to issuing another motion command.

Following a call to **MCStop** on the DCX-PC100 controller when in velocity mode, call **MCSetOperatingMode** prior to issuing another motion command.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

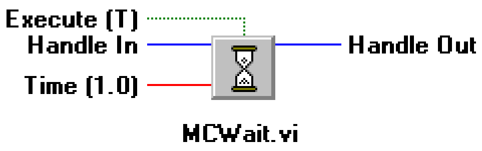
MCCL Reference

ST

See Also

MCAbort, MCEnableAxis, **MCSetOperatingMode**, **MCWaitForStop**

MCWait



MCWait waits the specified number of seconds before allowing the next Motion VI to execute.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

DBL **Time** selects the wait time, in seconds.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

Comments

The delay is specified in seconds, unless **MCSetScale** has been called to change the time scale.

> ⚠ Once this command is issued, no other VIs will be able to communicate with the board until **Time** elapses. We recommend creating your own time based looping structure.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

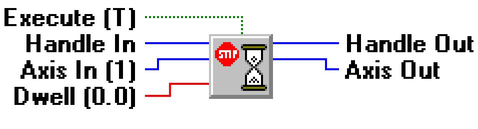
MCCL Reference

WA

See Also

MCWaitForStop

MCWaitForStop



MCWaitForStop.vi

MCWaitForStop waits for the specified axis to come to a stop. An optional dwell after the stop may be specified within this VI to allow the mechanical system to come to rest.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to abort the motion of.

DBL **Dwell** selects the dwell time, in seconds.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

MCWaitForStop is necessary for synchronizing motions, and for making certain that a prior motion has completed before beginning a new motion.

> ⚠ Once this VI is executed, no other VIs will be able to communicate with the board until **Axis In** reaches its target. We recommend using **MCGetStatus** / **MCDecodeStatus** to test for TRAJECTORY COMPLETE.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

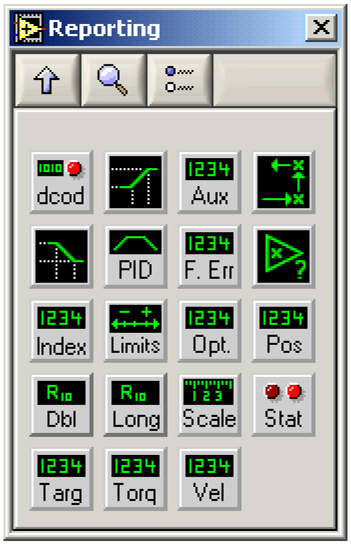
MCCL Reference

WS

See Also

MCWait

Chapter Contents



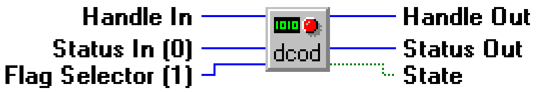
Chapter

7

Reporting VIs

Reporting VIs allow the calling program to query the board to determine how parameters have been configured, as well as getting information regarding the position and status of any given axis. These VIs may be used to read motor position, programmed velocity, PID filter settings, scale factors, status, and more.

MCDecodeStatus



MCDecodeStatus.vi

The **MCDecodeStatus** VI permits you to test flags in the controller status word in a way that is independent of the type of controller being inspected.

Parameters

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U32 **Status In** is the controller status word to decode.

I16 **Flag Selector** selects the status information to decode.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U32 **Status Out** is an output copy of the status word.

Comments

Using this function to test the status word returned by **MCGetStatus** isolates the program from controller dependent bit ordering of the status word. Please see the description of the **MCDecodeStatus** function in the online Motion Control API (MCAPI) Reference for specific information about the **Flag Selector** value.

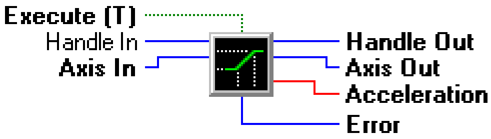
Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

See Also

MCGetStatus, online help sample programs

MCGetAccelerationEx



MCGetAccelerationEx.vi

The return value is the programmed acceleration of the axis selected.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the acceleration of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Acceleration** is the current acceleration value for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Compatibility

The DC2 stepper axes do not support ramping.

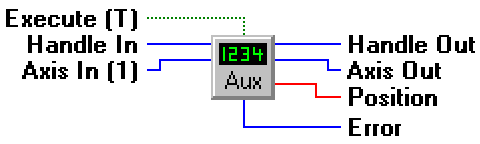
Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

See Also

MCSetAcceleration

MCGetAuxEncPosEx



MCGetAuxEncPosEx.vi

This VI returns the current auxiliary encoder position, if the axis supports auxiliary encoders.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the auxiliary encoder position of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Position** is the auxiliary encoder position for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

The auxiliary encoder's position may be set using the **MCSetAuxEncPos** VI.

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Compatibility

The DC2, DCX-PCI100 controllers, MC100, MC110, MC150, and MC320 modules do not support auxiliary encoders. Closed-loop steppers do not support auxiliary encoder functions, since the connected encoder is considered a primary encoder.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.1 or higher

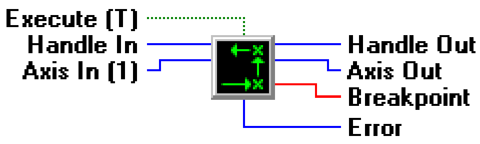
MCCL Reference

AT

See Also

MCSetAuxEncPos

MCGetBreakpointEx



MCGetBreakpointEx.vi

This VI returns the current breakpoint position

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the breakpoint of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Breakpoint** is the next breakpoint setting for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Compatibility

The DCX-PC100 controller and stepper axes do not support this command.

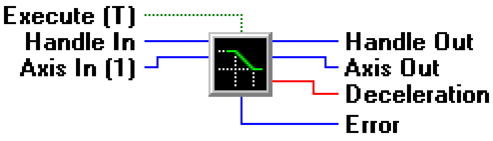
Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

MCCL Reference

TB

MCGetDecelerationEx



MCGetDecelerationEx.vi

This VI returns the current programmed deceleration value for the given axis, in whatever units the axis is configured for.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the deceleration of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Deceleration** is the current deceleration value for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

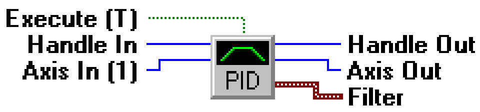
MCCL Reference

Controller RAM Motor Tables

See Also

MCSetDeceleration

MCGetFilterConfig



MCGetFilterConfig.vi

MCGetFilterConfig obtains the current PID filter configuration contained in the **Filter** cluster for the specified axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the PID filter settings from.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Filter is a cluster containing the current PID filter settings for the axis. The **Filter** cluster contains the following values:

DBL	**Derivative Gain** sets the derivative term of the PID loop.
DBL	**DerSample Period** is the time interval, in seconds, between derivative samples.
DBL	**Integral Gain** sets the integral term of the PID loop.
DBL	**Integration Limit** limits the power the integral gain can use to reduce error to zero.
DBL	**Velocity Gain** sets the feed-forward gain of the PID loop, volts per encoder count per second.
DBL	**Acceleration Gain** sets the feed-forward acceleration gain setting.
DBL	**Deceleration Gain** sets the feed-forward deceleration gain setting.
DBL	**Following Error** is the maximum position error, default units are encoder counts.

Comments

You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Compatibility

Velocity Gain is not supported on the DCX-PCI100 controller, MC100, MC110 modules, or closed-loop steppers. **Acceleration Gain** is not supported on the DC2, DCX-PC100, and DCX-PCI100 controllers. **Deceleration Gain** is not supported on the DC2, DCX-PC100, and DCX-PCI100 controllers.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

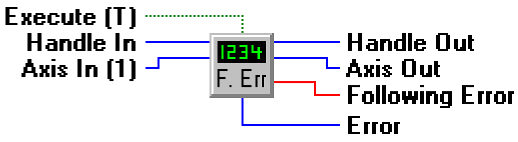
MCCL Reference

TD, TF, TG, TI, TL, Controller RAM Motor Tables

See Also

MCSetFilterConfig

MCGetFollowingError



MCGetFollowingError.vi

MCGetFollowingError returns the current following error (difference between the actual and the optimal positions) for the specified axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the following error of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Following Error** is the current following error value for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

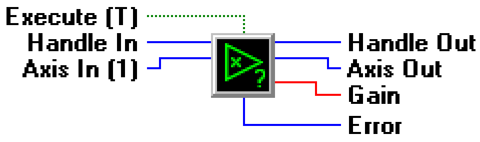
MCCL Reference

TF

See Also

MCGetOptimalEx, **MCGetPositionEx**

MCGetGain



MCGetGain.vi

MCGetGain returns the current gain setting for the specified axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the gain of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Gain** is the current gain value for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

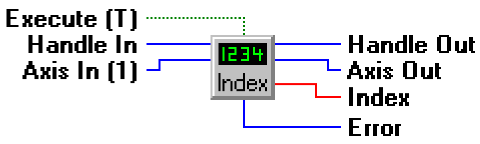
MCCL Reference

TG

See Also

MCSetGain

MCGetIndexEx



MCGetIndexEx.vi

The return value is the index position of the axis selected.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the index of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Index** is the current index value for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

Controller resets and the **MCSetPosition** VI may change the position reading of the primary encoder.

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Compatibility

The MC100, MC110 modules, and all stepper axes do not support this function.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

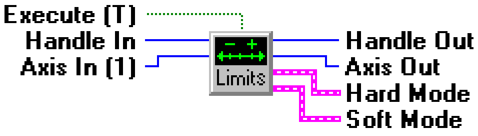
MCCL Reference

TZ

See Also

MCSetPosition

MCGetLimits



MCGetLimits.vi

MCGetLimits obtains the current limit settings for the specified axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the limit settings from.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

▣ **Hard Mode** is a cluster containing the current hard limit settings for the axis. The cluster values are as follows:

TF	**Low Limit** set to TRUE enables the lower hard limit.
TF	**High Limit** set to TRUE enables the upper hard limit.
U16	**Mode** selects stop mode – Turn Off (0) / Abrupt (4) / Smooth (8)

▣ **Soft Mode** is a cluster containing the current soft limit settings for the axis. The cluster values are as follows:

TF	**Low Limit** set to TRUE enables the lower soft limit.
TF	**High Limit** set to TRUE enables the upper soft limit.
U16	**Mode** selects stop mode – Turn Off (0) / Abrupt (4) / Smooth (8)
DBL	**Low Set** sets the lower soft limit value.
DBL	**High Set** sets the upper soft limit value.

Comments

You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Compatibility

The DC2 and DCX-PC100 controllers do not support soft limits.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.1 or higher

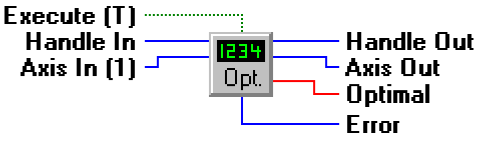
MCCL Reference

HL, LF, LL, LM, LN, Controller RAM Motor Tables

See Also

MCSetLimits

MCGetOptimalEx



MCGetOptimalEx.vi

Optimal returns the optimal position of the axis selected.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the optimal position from.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Optimal** is the current optimal position value for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

The trajectory generator generates an optimal position based upon an ideal (i.e. error free) motor. The PID loop then compares the actual position to the optimal position to calculate a correction to the actual trajectory. The maximum difference allowed between the optimal and actual positions is set with **Following Error** cluster member of the **MCSetFilterConfig** VI.

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Compatibility

The DC2 stepper axes do not support this command.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

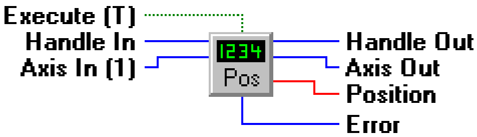
MCCL Reference

TO

See Also

MCGetFilterConfig, **MCSetFilterConfig**, **MCSetPosition**

MCGetPositionEx



MCGetPositionEx.vi

Position returns the value of the current position of the axis specified.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the position of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Position** is the current position value for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

> i You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

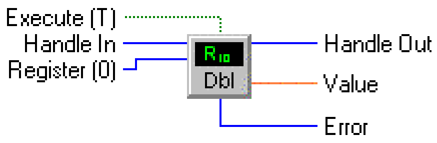
MCCL Reference

TP

See Also

MCSetPosition, **MCSetScale**

MCGetRegisterDouble



MCGetRegisterDouble.vi

The **MCGetRegisterDouble** VI reads the value of a motion controller double precision floating point register.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Register** selects the register to read the value of.

DBL **Value** is the value read from the register.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

When running background tasks on a multitasking controller the only way to communicate with the background tasks is to pass parameters in the general purpose registers. You cannot write to the local registers (registers 0 - 9) of a background task. When you need to communicate with a background task be sure to use one or more of the global registers (10 - 255).

Requirements

MCAPI: version 2.0 or higher
Motion VI Library: version 2.0 or higher

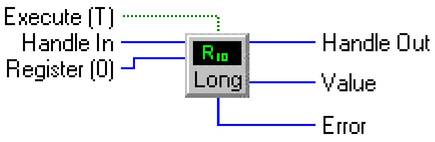
MCCL Reference

TR

See Also

MCGetRegisterLong, **MCSetRegisterDouble**, **MCSetRegisterLong**

MCGetRegisterLong



MCGetRegisterLong.vi

The **MCGetRegisterLong** VI reads the value of a motion controller 32-bit integer register.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Register** selects the register to read the value of.

I32 **Value** is the value read from the register.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

When running background tasks on a multitasking controller the only way to communicate with the background tasks is to pass parameters in the general purpose registers. You cannot write to the local registers (registers 0 - 9) of a background task. When you need to communicate with a background task be sure to use one or more of the global registers (10 - 255).

Requirements

MCAPI: version 2.0 or higher
Motion VI Library: version 2.0 or higher

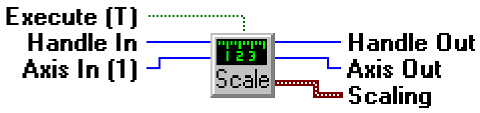
MCCL Reference

TR

See Also

MCGetRegisterDouble, **MCSetRegisterDouble**, **MCSetRegisterLong**

MCGetScale



MCGetScale.vi

MCGetScale obtains the current scaling factors for the specified axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the acceleration of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Scaling is a cluster containing the current scale factors for the axis.

DBL	**Constant** acts as a scale factor for servo analog outputs. By calibrating your motor/amplifier combination, it is possible to scale the output with **Constant** so that torque settings may be specified directly in ft-lbs.
DBL	**Offset** represents an offset from a servo encoder's index pulse to a zero position.
DBL	**Rate** acts as a multiplier for motion commands time values. The base controller time unit is the second, to convert this to minutes set **Rate** to 60.0, to convert to milliseconds rate should be set to 0.001
DBL	**Scale** is applied to motion parameters to convert from encoder counts to real world units.
DBL	**Zero** specifies that a soft zero should be located this distance from actual zero. By moving the soft zero around it is possible to have a series of position commands repeated at various spots in the range of travel without modifying the position commands. The actual zero position is not changed by this command.

DBL	**Time** is the time factor for controller level wait commands. See the discussion of the **Rate** parameter above for more information on setting this value. Note that a single **Time** value is maintained per controller (i.e. **Time** is axis independent).

Comments

Scaling allows the application to communicate with the controller in real world units such as inches, meters, and radians; as opposed to low level (i.e. un-scaled) values such as raw encoder counts, etc.

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Compatibility

The DC2 and DCX-PC controllers do not support scaling.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

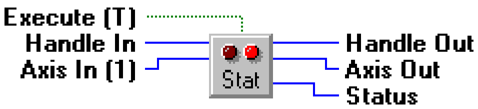
MCCL Reference

Controller RAM Motor Tables

See Also

MCSetScale

MCGetStatus



MCGetStatus.vi

MCGetStatus returns the controller dependent status word for the specified axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the status of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I32 **Status** is the current status word for axis.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

Use the **MCDecodeStatus** VI to test specific flags in the status word.

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

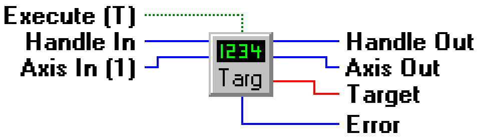
MCCL Reference

TS

See Also

MCDecodeStatus, Controller hardware reference manual

MCGetTargetEx



MCGetTargetEx.vi

Target returns the value of the target position of the axis selected.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the target position of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Target** is the current target position value for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

The VIs **MCMoveAbsolute** and **MCMoveRelative** update the target position for an axis. The controller will then generate an optimal trajectory to the target position, and the PID loop will seek to minimize the error (difference between actual and optimal trajectories).

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

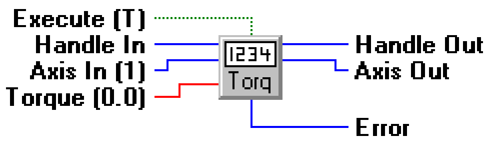
MCCL Reference

TT

See Also

MCMoveAbsolute, **MCMoveRelative**

MCGetTorque



MCSetTorque.vi

MCGetTorque returns the current torque setting for the specified axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the torque of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Torque** is the current torque value for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

> i You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Compatibility

Torque mode is not supported on stepper axes, DCX-PCI100 controller, MC100, or MC110 modules.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

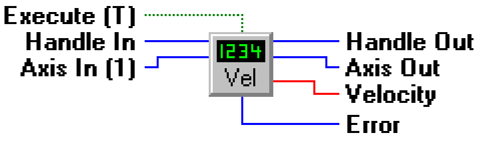
MCCL Reference

TQ

See Also

MCSetTorque

MCGetVelocityEx



MCGetVelocityEx.vi

The return value is the programmed velocity of the axis selected.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to get the velocity of.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

DBL **Velocity** is the current velocity value for axis.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

> You may not set the **Axis In** parameter to ALL AXES (a value of zero) for this VI.

Requirements

MCAPI: version 1.3 or higher
Motion VI Library: version 1.0 or higher

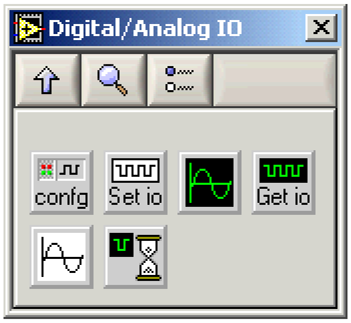
MCCL Reference

Controller RAM Motor Tables

See Also

MCSetVelocity

Chapter Contents



Chapter

8

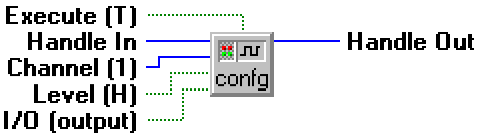
Analog & Digital I/O VIs

This section describes the VIs for control of the on-board, undedicated digital and analog I/O channels. These VIs configure the operation of, check the state of, and change the state of the on-board I/O channels. The number and type of I/O channels varies with the type of controller, and with the number and type of installed modules.

Digital I/O VIs allow configuration of high or low “true” states, reading of inputs, sequencing based on input, and setting outputs. Analog I/O VIs control the input and output of analog values through A/D and D/A ports if installed on the controller.

A word of caution must be given regarding the use of board-level sequencing commands. Even though a warning is included with **MCWaitForDigitalIO,** it should be stressed that once this VI executes, no other VI will be able to communicate with the motion control card nor will card respond until it has completed what it was initially told to do. This can lead to scenarios where the calling program has absolutely no control during potentially dangerous or otherwise expensive situations.

MCConfigureDigitalIO



MCConfigureDigitalIO.vi

Configures a digital channel for input or output, and sets the logic level to high true logic or low true logic.

Parameters

TF	**Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.
I16	**Handle In** is the controller handle returned by the MCOpen VI.
U16	**Channel** is the channel number (between 1 and the total number of installed digital I/O channels) to configure.
TF	**Level** sets the logic level to high true logic if TRUE (the default), or low true logic if FALSE.
TF	**I/O** should be set to TRUE (the default) to configure the channel for output, or FALSE to configure the channel for input.
I16	**Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

Comments

Most of the digital I/O channels may be configured for input or for output. The logic level maps the logical "on" and "off" states of the channel to the physical input and output voltages for that channel. If the channel is set to MC_DIO_LOW (negative logic) the "on" state of a channel will represent a low voltage (<0.4VDC) and "off" a high voltage (>2.4VDC). When set to MC_DIO_HIGH (positive logic) the "on" state of a channel will represent a high voltage (>2.4VDC) and "off" a low voltage (<.0.4VDC).

The DCX-PCI motherboard has 16 general I/O, consisting of 8 fixed inputs and 8 fixed outputs. Since these digital I/O are fixed, they may not be configured for input or output. A program may verify the functionality (input or output) of a channel by using **MCGetDigitalIOConfig** to check the current configuration.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

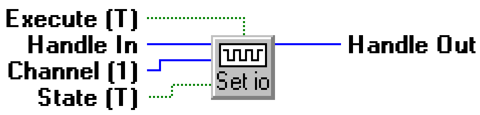
MCCL Reference

CH, CI, CL, CT

See Also

MCEnableDigitalIO, **MCGetDigitalIO**

MCEnableDigitalIO



MCEnableDigitalIO.vi

This VI function turns the specified digital I/O channel on or off, depending upon the value of **State**.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Channel** is the channel number (between 1 and the total number of installed digital I/O channels) to enable or disable.

TF **State** should be set to TRUE (the default) to enable the channel, or FALSE to disable the channel.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

Comments

The I/O channel selected must have previously been configured for output using the **MCConfigureDigitalIO** VI. Note that depending upon how a channel has been configured "on" (and conversely "off") may represent either a high or a low voltage level.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

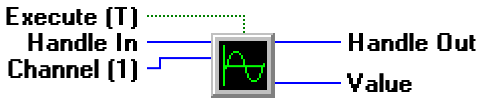
MCCL Reference

CF, CN

See Also

MCConfigureDigitalIO, **MCGetDigitalIO**

MCGetAnalog



MCGetAnalog.vi

MCGetAnalog reads the current input state of the specified input **Channel**.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Channel** is the channel number (between 1 and the total number of installed analog input channels) to read from.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I16 **Value** is the digitized reading from the analog input channel.

Comments

The DC2, DCX-AT, and DCX-PC controllers all include four undedicated 8-bit analog input channels. By default these channels are assigned channel numbers 1 to 4. Each analog input accepts an input voltage between 0 and +5 volts. The value read in from the channel will be the ratio of the input voltage to the reference voltage times 255. An internal 5.0 volt reference is supplied by the controller; an external reference may be supplied in place of the internal reference if desired.

$$value = \frac{V_{Input}}{V_{Reference}} x\ 255$$

Additional analog input/output channels supplied by MC500 modules will occupy sequential channel numbers beginning with channel 5.

Compatibility

There are no compatibility issues with this function, however, please note that the DCX-PCI controllers have no built-in analog inputs.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

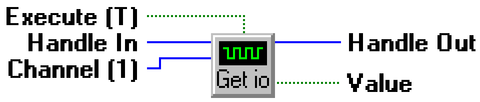
MCCL Reference

TA

See Also

MCSetAnalog

MCGetDigitalIO



MCGetDigitalIO.vi

The **MCGetDigitalIO** VI returns the current state of the specified digital I/O channel.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Channel** is the channel number (between 1 and the total number of installed digital I/O channels) to read from.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

TF **Value** is TRUE if the specified channel is on, or FALSE if the channel is off.

Comments

This function will read the current state of both input and output digital I/O channels. Note that this function simply reports if the channel is "on" or "off"; depending upon how a channel has been configured "on" (and conversely "off") may represent either a high or a low voltage level.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

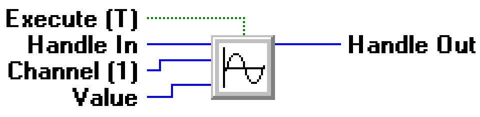
MCCL Reference

TC

See Also

MCConfigureDigitalIO, **MCEnableDigitalIO**

MCSetAnalog



MCSetAnalog.vi

MCSetAnalog sets the output level of an analog channel.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Channel** is the channel number (between 1 and the total number of installed analog output channels) to set.

U16 **Value** is the new output value for the specified output channel.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

Comments

Analog output ports on MC500 and MC520 Analog Modules accept values in the range of 0 to 4095 counts (12 bits). This range of values corresponds to an output voltage of 0 to 5V or -10 to +10V, depending upon how the output is configured (see your controller's hardware manual). Each digital bit corresponds to a voltage level as follows:

Output Used	Volts per Count
0 to 5V	0.0012V
-10 to +10V	0.0049V

Compatibility

Analog output channels are not supported by the DC2-PC100 dedicated 2 axis controllers.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

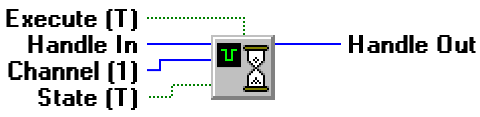
MCCL Reference

OA

See Also

MCGetAnalog

MCWaitForDigitalIO



MCWaitForDigitalIO.vi

MCWaitForDigitalIO waits for the specified digital I/O channel to go on or off, depending upon the value of **State**.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Channel** is the channel number (between 1 and the total number of installed digital I/O channels) to wait on.

TF **State** should be set to TRUE (the default) to wait for the channel to go on, or FALSE to wait for the channel to go off.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

Comments

Digital channels 1 to 16 are built into each controller. Additional digital channels, beginning with channel 17, may be added in blocks of 16 channels using MC400

> ⚠ Once this VI is executed, no other VIs will be able to communicate with the board until the digital I/O is equal to **State**. We recommend creating your own looping structure based on **MCGetDigitalIO** instead.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

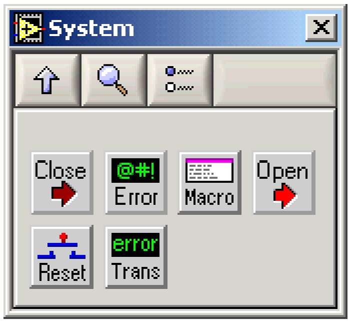
MCCL Reference

WF, WN

See Also

MCConfigureDigitalIO, **MCEnableDigitalIO**, **MCGetDigitalIO**

Chapter Contents



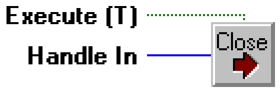
Chapter

9

System VIs

These VI's handle system level functions, including the opening and closing of a particular controller, and error handling. This library also contains the MCAPI / LabVIEW controls (handle in/out, axis in/out) used by the other VIs in that make up the MCAPI / LabVIEW components. You will not normally need to use these controls directly as they are already incorporated into the supplied VIs. They are available, however, if you wish to extend the library yourself.

MCClose



MCClose.vi

MCClose closes the specified motion controller handle, and is typically called at the end of a program.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

Comments

Following a call to **MCClose**, no further calls should be made to the Motion Control API functions with this handle (the exception being **MCOpen**, which may be called to open or reopen the API at any time).

By calling **MCClose** you notify Windows that you are done with the controller and device driver. When the last user has closed the driver Windows is then free to unload the driver from memory. Failure to call close leaves the handle open, reducing the number of available controller handles for other applications.

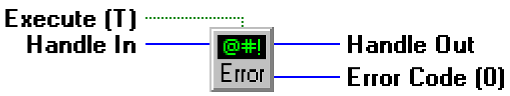
Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

See Also

MCOpen

MCGetError



MCGetError.vi

The **MCGetError** VI returns the most recent error code for **Handle In**.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I16 **Error Code** is the most recent numeric error code, or zero if there was no error.

Comments

The error is cleared after it has been read. Errors are maintained on a per-handle basis, calls to **MCGetError** only return errors that occurred during function calls that used the same handle.

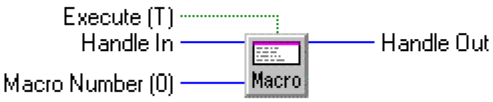
Requirements

MCAPI: version 1.2 or higher
Motion VI Library: version 1.0 or higher

See Also

MCTranslateErrorEx

MCMacroCall



MCMacroCall.vi

MCMacroCall executes a previously stored macro.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Macro Number** selects the macro number to execute.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

Comments

Macros are normally downloaded using the **MCPuts** ASCII interface command, using the Motion Control Command Language (MCCL). These controller level macros are often the only efficient way to implement hardware specific sequences, such as special homing routines, initializing encoder positions, etc.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 2.0 or higher

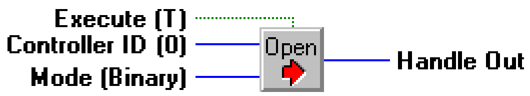
MCCL Reference

MC

See Also

MCPuts, Controller hardware manual

MCOpen



MCOpen.vi

MCOpen returns a handle to a particular controller for use with subsequent VI calls.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Controller ID** selects the controller to open. The ID selected must have been previously configured using the MCSetup program supplied with the MCAPI.

◂▸ **Mode** specifies the open mode - ASCII. ASCII EXCLUSIVE, BINARY (the default) or BINARY EXCLUSIVE.

I16 **Handle Out** is controller handle that is required by all other motion VIs.

Comments

This function returns handle to the specified controller for use in subsequent VIs. The handle will be greater than zero if the open call succeeds, or less than zero if there is an error. Standard error codes will be multiplied by -1 to make their values negative and returned in place of a handle if there is an error.

Always wire the handle returned by **MCOpen** and use that value in subsequent VIs. **MCOpen** must be executed before any other VIs are attempted. If **MCOpen** detects an error it will display an informative dialog box and abort execution of the program (source for **MCOpen** is included with the Motion VI Library so that you may modify this behavior). For details about specific error codes see the Error Code cross-reference.

If it is necessary that no one else gains access to a controller while you are using it, you may set the open mode to ASCII EXCLUSIVE or BINARY EXCLUSIVE.

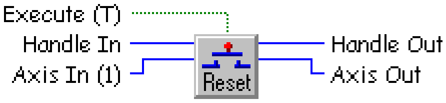
Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

See Also

MCClose

MCReset



MCReset.vi

MCReset performs a complete reset of the axis or controller, leaving the specified axis (or axes) in the disabled state.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to reset.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

Comments

Setting the **Axis In** parameter to ALL AXES (a value of zero) will cause the specified controller to be reset (including all installed axes).

If you have enabled the hardware reset feature of the DCX-AT, or DCX-PC100 controllers **MCReset** will perform a hard reset when **Axis In** is equal to ALL AXES (a value of zero), or a soft reset when **Axis In** specifies a particular axis. If this feature is off (the default state), **MCReset** issues the “RT” command to the board to perform any reset (this is a "soft" reset). On the DCX-AT200 and DCX-AT300 you must set jumper JP2 to connect pins 1 and 2 if Hard Reset is enabled, or connect pins 5 and 6 (factory default) if Hard Reset is disabled. On the DCX-PC100 you must set jumper JP4 to connect pins 1 and 2 if Hard Reset is enabled, or connect pins 5 and 6 (factory default) if Hard Reset is disabled. See the Motion Control Panel online help for how to enable the MCAPI Hardware Reset feature. At this time, the DCX-PCI controllers only support soft resets.

Compatibility

The DC2 series, DCX-PC100, DCX-AT100, and DCX-AT200 (prior to firmware version 1.2a) controllers do not support the resetting of individual axes. In these cases when this command is executed, the **Axis In** parameter is ignored and a controller reset is performed.

Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 2.0 or higher

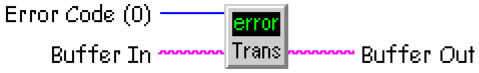
MCCL Reference

RT

See Also

MCAbort, **MCStop**

MCTranslateErrorEx



MCTranslateErrorEx.vi

MCTranslateErrorEx translates the numeric error code returned by a Motion VI into a readable string message.

Parameters

I16 **Error Code** is a numeric error code returned by one of the Motion VIs.

abc **Buffer In** is the string buffer that will hold the error message string. It is recommended that **Buffer In** be at least 64 characters long.

abc **Buffer Out** contains the error message string,

Comments

For details about specific error codes see the Error Codes table in Appendix A.

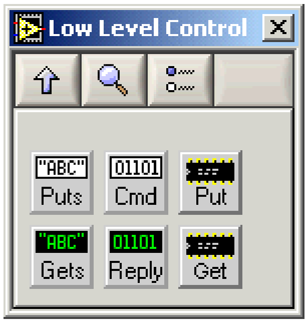
Requirements

MCAPI: version 2.1 or higher
Motion VI Library: version 2.0 or higher

See Also

MCGetError

Chapter Contents



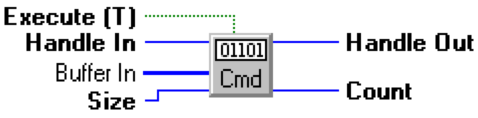
Chapter

10

Low-Level OEM VIs

These VI's provide a low-level interface to the motion control card. They permit you to read and write ASCII (text) or binary commands directly to the motion control card. See your control card hardware reference manual for more information.

MCCommand



MCCommand.vi

The **MCCommand** VI downloads a formatted binary command buffer directly to the controller.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

[U8] **Buffer In** is a formatted binary command buffer. See your motion control card hardware reference for more information.

I16 **Size** specifies the size of the command buffer, in bytes.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I16 **Count** is the actual number of bytes transmitted.

Comments

The return value from this VI is the actual number of bytes downloaded. Because of the nature of the binary interface, the return value will be equal to either the buffer size (value of the ***Size*** argument), indicating the command buffer was successfully downloaded, or zero, indicating a problem communicating with the controller.

The binary interface is described in detail in the hardware manual that accompanied your controller. The user of this VI is responsible for correctly formatting the buffer - no checking is performed by the VI.

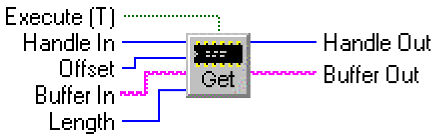
Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

See Also

MCReply

MCGetRam



MCGetRam.vi

The **MCGetRam** function reads **Length** bytes from controller memory beginning at memory location **Offset**.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Offset** specifies the starting controller memory address to read from.

abc **Buffer In** is a buffer for the values read from controller memory. This buffer must be at least **Length** bytes long!

I16 **Length** specifies the size of the **Buffer**, in bytes.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

abc **Buffer Out** is the filled buffer.

Comments

No range checking is performed on **Offset** or **Length** - it is the user’s responsibility to supply valid values for these arguments. Consult your controller hardware manual for details on the controller memory map.

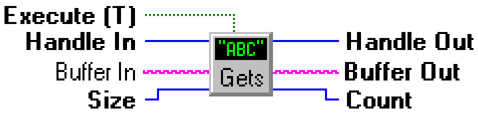
Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

See Also

MCPutRam

MCGets



MCGets.vi

The **MCGets** VI reads a null-terminated ASCII string of up to **Size** characters from the controller ASCII interface.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

abc **Buffer In** is pre-allocated buffer, large enough to hold the reply from the control card.

I16 **Size** specifies the maximum number of characters allowed in the output buffer.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

abc **Buffer Out** is the ASCII reply from the control card.

I16 **Count** is the actual number of bytes read.

Comments

The return value from this VI is number of bytes actually read from the controller. This VI will wait for a reply for as long as the controller is busy processing commands and will only return a zero when the controller is idle and there are no reply characters.

> You must open the controller in ASCII mode (MC_OPEN_ASCII) in order to use this command.

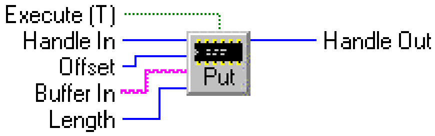
Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

See Also

MCPuts

MCPutRam



MCPutRam.vi

The **MCPutRam** VI writes **Length** bytes into controller memory beginning at memory location **Offset**.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Offset** specifies the beginning location in controller memory to write the data to.

abc **Buffer In** contains the data to be written into controller memory. This buffer must be at least **Length** bytes long!

I16 **Length** specifies the size of the **Buffer**, in bytes.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

Comments

Consult your controller hardware manual for details on the controller memory map.

⚠ No range checking is performed on **Offset** or **Length**. It is the caller's responsibility to supply valid values for these arguments. Writing directly to dual ported ram can cause unpredictable results. **USE THIS FUNCTION WITH EXTREME CAUTION!**

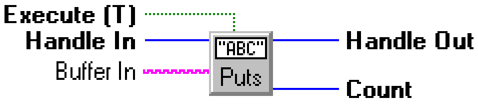
Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

See Also

MCGetRam

MCPuts



MCPuts.vi

The **MCPuts** VI writes a NULL terminated command string to the controller ASCII interface.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

abc **Buffer In** is a null-terminated ASCII command string. See your motion control card hardware reference for more information.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I16 **Count** is the actual number of bytes transmitted.

Comments

This VI returns the number of characters actually written to the controller. This number may be less than the length of the string if the controller becomes busy and stops accepting characters.

Remember to include a carriage return "\r" in all command strings in order for the command to be executed.

> You must open the controller in ASCII mode (MC_OPEN_ASCII) in order to use this command.

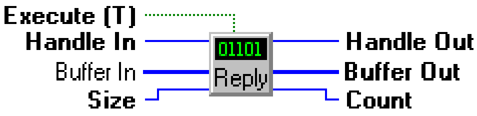
Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

See Also

MCGets

MCReply



MCReply.vi

The **MCReply** VI reads a binary reply of up to **Size** bytes from the controller.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

[U8] **Buffer In** is a pre-allocated buffer for the reply.

I16 **Size** specifies the size of the reply buffer, in bytes.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

[U8] **Buffer Out** is a buffer containing the binary reply from the control card.

I16 **Count** is the actual number of bytes received from the motion control card.

Comments

The return value from this VI is the actual number of bytes read. This value may be less than the argument **Size**, but it will never exceed **Size**. If the controller has no reply ready the return value will be zero.

This VI waits for a reply for as long as the controller is busy - it returns with a return value of zero if no reply is (or will be) available.

> You must open the controller in ASCII mode (MC_OPEN_ASCII) in order to use this command.

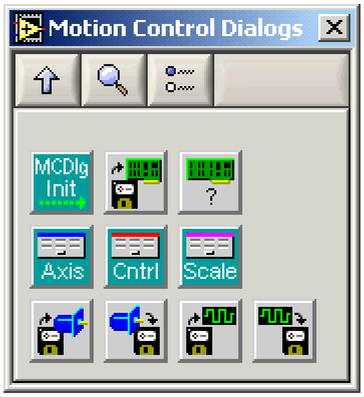
Requirements

MCAPI: version 1.0 or higher
Motion VI Library: version 1.0 or higher

See Also

MCCommand

Chapter Contents



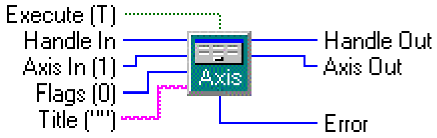
Chapter

11

Motion Dialog VIs

The Common Motion Dialog library includes easy-to-use VIs for the control and configuration of your motion controller. By combining these functions in a single library we've made it easy for programmers to include the Common Motion Dialog functionality in their application programs. VIs are provided for the configuration of servo and stepper axes, scaling setup, controller selection, file download, and save/restore of motor settings.

MCDLG_ConfigureAxis



MCDLG_ConfigureAxis.vi

MCDLG_ConfigureAxis displays a servo or stepper axis setup dialog that permits user configuration of the axis.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to configure.

abc **Title** specifies an optional title for the dialog box. Leave blank to use the default value.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

This VI invokes a comprehensive, ready-to-use setup dialog for stepper and servo motor axis types. The dialog initializes itself by querying the motion controller for the current axis settings. Any changes the user makes are sent to the motion controller if the user dismisses the dialog by pressing the OK button.

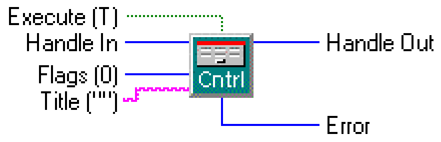
Requirements

MCAPI: version 2.1 or higher
Motion VI Library: version 2.0 or higher

See Also

MCDLG_Initialize, **MCDLG_RestoreAxis**, **MCDLG_SaveAxis**

MCDLG_ControllerInfo



MCDLG_ControllerInfo.vi

MCDLG_ControllerInfo displays configuration information about the specified motion controller.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

abc **Title** specifies an optional title for the dialog box. Leave blank to use the default value.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

This VI displays a read only dialog providing information on the current motion controller configuration and capabilities (this information is typically used by programs to control execution - can the controller multi-task? Is contouring supported?).

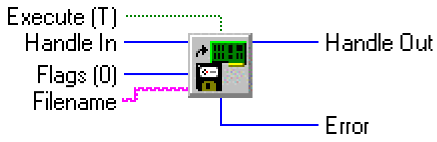
Requirements

MCAPI: version 2.1 or higher
Motion VI Library: version 2.0 or higher

See Also

MCDLG_Initialize

MCDLG_DownloadFile



MCDLG_DownloadFile.vi

The **MCDLG_DownloadFile** VI opens the specified file and downloads the contents to the specified controller.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

abc **Filename** specifies the name of the command file to be downloaded (this filename may include a drive letter and path).

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

If you routinely configure a motion controller with macros this function makes it easy to download those macros to the motion controller (simply save the macros in a text file and pass the name of that file to this VI).

> The handle passed to this VI must have been opened in ASCII mode.

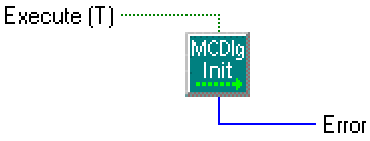
Requirements

MCAPI: version 2.1 or higher
Motion VI Library: version 2.0 or higher

See Also

MCDLG_Initialize

MCDLG_Initialize



MCDLG_Initialize.vi

MCDLG_Initialize must be called before any other MCDLG VIs are used.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

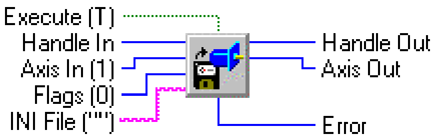
Comments

Using **MCDLG_Initialize** ensures that internal data structures in the MCDLG Library are correctly initialized.

Requirements

MCAPI: version 2.1 or higher
Motion VI Library: version 2.0 or higher

MCDLG_RestoreAxis



MCDLG_RestoreAxis.vi

The **MCDLG_RestoreAxis** VI restores the settings of the given axis to a previously saved state.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to restore settings for. This value may be set to 0 (all axes) to restore settings for all axes on a particular controller.

I32 **Flags** may be set to a non-zero value to selectively disable the restoring of certain groups of settings. Leave this value set to zero (the default) to restore all settings. See the MCDLG Reference (included with the Motion Control API) online help for details of the values for **Flags**.

abc **INI File** specifies the name of the INI file to retrieve the settings from. Leave this string blank to use the default file (MCAPI.INI).

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

MCDLG_SaveAxis encodes the motion controller type and module type into signature that is saved with the axis settings. **MCDLG_RestoreAxis** checks for a valid signature before restoring the axis settings. If you make changes to your hardware configuration (i.e. change module types or controller type) **MCDLG_RestoreAxis** will refuse to restore those settings.

You may specify ALL AXES (a value of zero) for the **Axis In** parameter in order to restore the parameters for all axes installed on a motion controller with a single call to this function.

Restoring the parameters to an axis while it is moving may result in erratic behavior (such as when you choose to include the motor position in the restored parameters). The flag MCDLG_CHECKACTIVE (a value of 2048) forces this function to check each restored axis to see if it is active before it proceeds. By default MCDLG_CHECKACTIVE (a value of 2048) will skip the restore of an active axis, but if you also include the flag MCDLG_PROMPT (a value of 1), which would yield a flag of 2049, the user will be prompted for how to proceed.

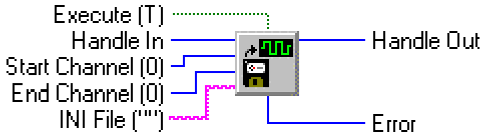
Requirements

MCAPI: version 2.1 or higher
Motion VI Library: version 2.0 or higher

See Also

MCDLG_Initialize, **MCDLG_SaveAxis**

MCDLG_RestoreDigitalIO



MCDLG_RestoreDigitalIO.vi

MCDLG_RestoreDigitalIO restores the settings of the all the digital I/O channels between **Start Channel** and **End Channel** (inclusive) to their previously saved states.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Start Channel** specifies the starting channel number to restore settings for. This value may be set to 0 to specify the first channel on a controller.

U16 **End Channel** specifies the ending channel number to restore settings for. This value may be set to 0 to specify the last channel on a controller.

abc **INI File** specifies the name of the INI file to retrieve the settings from. Leave this string blank to use the default file (MCAPI.INI).

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

By setting **Start Channel** and **End Channel** both to zero this VI will automatically restore all the digital I/O channels on a motion controller.

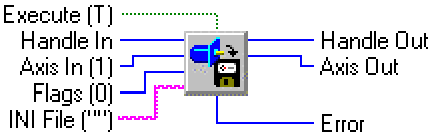
Requirements

MCAPI: version 2.1 or higher
Motion VI Library: version 2.0 or higher

See Also

MCDLG_Initialize, **MCDLG_SaveDigitalIO**

MCDLG_SaveAxis



MCDLG_SaveAxis.vi

The **MCDLG_SaveAxis** VI saves the settings of the given axis, allowing them to be restored at a latter time.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to save settings for. This value may be set to 0 (all axes) to save settings for all axes on a particular controller.

I32 **Flags** may be set to a non-zero value to selectively disable the saving of certain groups of settings. Leave this value set to zero (the default) to save all settings. See the MCDLG Reference (included with the Motion Control API) online help for details of the values for **Flags**.

abc **INI File** specifies the name of the INI file to save the settings to. Leave this string blank to use the default file (MCAPI.INI).

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

MCDLG_SaveAxis encodes the motion controller type and module type into signature that is saved with the axis settings. **MCDLG_RestoreAxis** checks for a valid signature before restoring the axis settings. If you make changes to your hardware configuration (i.e. change module types or controller type) **MCDLG_RestoreAxis** will refuse to restore those settings.

You may specify the constant ALL AXES (a value of zero) for the **Axis In** parameter in order to save the parameters for all axes installed on a motion controller with a single call to this function. Setting

Axis In to -1 will cause **MCDLG_SaveAxis** to delete all of the stored axis information for this controller.

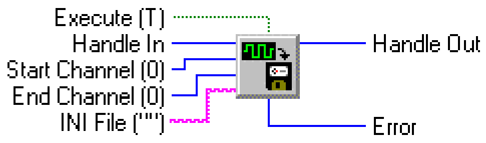
Requirements

MCAPI: version 2.1 or higher
Motion VI Library: version 2.0 or higher

See Also

MCDLG_Initialize, **MCDLG_RestoreAxis**

MCDLG_SaveDigitalIO



MCDLG_SaveDigitalIO.vi

MCDLG_SaveDigitalIO saves the settings of the all the digital I/O channels between **Start Channel** and **End Channel** (inclusive) to a file, allowing them to be easily restored to those settings at a latter time.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Start Channel** specifies the starting channel number to save settings for. This value may be set to 0 to specify the first channel on a controller.

U16 **End Channel** specifies the ending channel number to save settings for. This value may be set to 0 to specify the last channel on a controller.

abc **INI File** specifies the name of the INI file to retrieve the settings from. Leave this string blank to use the default file (MCAPI.INI).

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

By setting **Start Channel** and **End Channel** both to zero this function will automatically save all the digital I/O channels on a motion controller.

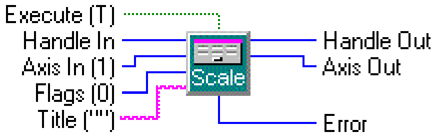
Requirements

MCAPI: version 2.1 or higher
Motion VI Library: version 2.0 or higher

See Also

MCDLG_Initialize, **MCDLG_RestoreDigitalIO**

MCDLG_Scaling



MCDLG_Scaling.vi

MCDLG_Scaling displays a scaling setup dialog and, if the motion controller supports scaling, allows the user to change the scaling parameters.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Handle In** is the controller handle returned by the MCOpen VI.

U16 **Axis In** selects the axis number to set the scaling of.

I32 **Flags** may be set to a non-zero value to change the behavior of the scaling dialog. Leave this value set to zero for the default behavior. See the MCDLG Reference (included with the Motion Control API) online help for details of the values for **Flags**.

abc **Title** specifies an optional title for the dialog box. Leave blank to use the default value.

I16 **Handle Out** is an output copy of the **Handle In** value, allowing motion VIs to be easily cascaded.

U16 **Axis Out** is an output copy of the **Axis In** value, allowing motion VIs to be easily cascaded.

I32 **Error** is zero if there were no errors or a non-zero error code if there was an error.

Comments

For controllers that don't support scaling the Motion Control API will fill in default values (zero for offsets, one for factors). **MCDLG_Scaling** will display these defaults as read-only. For advanced controllers such as the DCX-AT and the DCX-PCI **MCDLG_Scaling** will display the current scale factors and allow the user to change them.

Scaling changes will take effect following the next motor on command (**MCEnableAxis**) after **MCDLG_Scaling** completes.

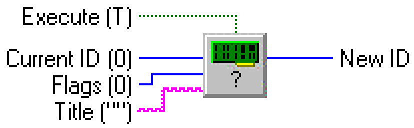
Requirements

MCAPI: version 2.1 or higher
Motion VI Library: version 2.0 or higher

See Also

MCDLG_Initialize

MCDLG_SelectController



MCDLG_SelectController.vi

MCDLG_SelectController displays a list of installed controllers and allows the user to select a controller from the list.

Parameters

TF **Execute** specifies whether the VI should execute or skip execution. The default value for execute is TRUE, allowing the VI to execute normally. This input may be wired to a Boolean switch to control the VI's execution. See the discussion of the Execute Input in Chapter 3 for more information.

I16 **Current ID** is the ID of the currently selected controller (set to –1 to ignore). The controller matching this ID will be pre-selected when the dialog first appears

U16 **Axis In** selects the axis number to set the scaling of.

I32 **Flags** is not currently supported and should be left blank.

abc **Title** specifies an optional title for the dialog box. Leave blank to use the default value.

I16 **New ID** is the ID of the controller selected by the user, or –1 if no controller was selected (or available). You must use the MCOpen VI with this value in order to use this controller.

Comments

This VI displays a list of installed controllers and allows the user to select one from the list. If a valid ID is given for **Current ID** that controller will be highlighted in the list as the default selection (set **Current ID** to -1 prevent a default selection). If no motion controllers have been configured for use with the Motion Control Applet in the Motion Control Panel, a message is displayed indicating that no controllers are configured and **New ID** will be set to -1.

Requirements

MCAPI: version 2.1 or higher
Motion VI Library: version 2.0 or higher

See Also

MCDLG_Initialize

Appendix

A

Error Codes

Motion VI Library error messages are listed numerically in the table below. Where possible corrective action has been included in the column labeled *Description*.

Error	Constant	Description
0	MCERR_NOERROR	No error has occurred.
1	MCERR_NO_CONTROLLER	No controller assigned at this ID. Use MCSETUP to configure a controller.
2	MCERR_OUT_OF_HANDLES	MCAPI driver out of handles. The driver is limited to 32 open handles. Applications that do not call **MCClose** when they exit may leave handles unavailable, forcing a reboot.
3	MCERR_OPEN_EXCLUSIVE	Cannot open - another application has the controller opened for exclusive use.
4	MCERR_MODE_UNAVAIL	Controller already open in different mode. Some controller types can only be open in one mode (ASCII or binary) at a time.
5	MCERR_UNSUPPORTED_MODE	Controller doesn't support this mode for **MCOpen** - i.e. ASCII or binary.
6	MCERR_INIT_DRIVER	Couldn't initialize the device driver.
7	MCERR_NOT_PRESENT	Controller hardware not present.
8	MCERR_ALLOC_MEM	Memory allocation error. This is an internal memory allocation problem with the DLL, contact Technical Support for assistance.
9	MCERR_WINDOWSERROR	A windows function returned an error - use **GetLastError( )** under WIN32 for details
10	-	reserved
11	MCERR_NOTSUPPORTED	Controller doesn't support this feature.
12	MCERR_OBSOLETE	Function is obsolete.
13	MCERR_CONTROLLER	Invalid controller handle.
14	MCERR_WINDOW	Invalid window handle.
15	MCERR_AXIS_NUMBER	Axis number out of range.
16	MCERR_AXIS_TYPE	Axis type doesn't support this feature.
17	MCERR_ALL_AXES	Cannot use MC_ALL_AXES for this function.
18	MCERR_RANGE	Parameter was out of range.
19	MCERR_CONSTANT	Constant value inappropriate.
20	MCERR_UNKNOWN_REPLY	Unexpected or unknown reply.
21	MCERR_NO_REPLY	Controller failed to reply.
22	MCERR_REPLY_SIZE	Reply size incorrect.
23	MCERR_REPLY_AXIS	Wrong axis for reply.
24	MCERR_REPLY_COMMAND	Reply is for different command.
25	MCERR_TIMEOUT	Controller failed to respond.
26	MCERR_BLOCK_MODE	Block mode error. Caused by calling **MCBlockEnd( )** without first calling **MCBlockBegin( )** to begin the block.
27	MCERR_COMM_PORT	Communications port (RS232) driver reported an error.

Error	Constant	Description
28	MCERR_CANCEL	User canceled action (such as when an MCDLG dialog box is dismissed with the CANCEL button.
29	MCERR_NOT_INITIALIZED	Feature was not correctly initialized before being enable or used.

Appendix B

Printing a PDF Document

Introduction to PDF
PDF stands for Portable Document Format. It is the de facto standard for transporting electronic documents. PDF files are based on the PostScript language imaging model. This enables sharp, color-precise printing on almost all printers.

Printing a complete PDF document
It is **not recommended** that large PDF documents be printed on personal computer printers. The 'wear and tear' incurred by these units, coupled with the difficulties of two sided printing, typically resulting in degraded performance of the printer and a whole lot of wasted paper. PMC recommends that PDF document be printer by a full service print shop that uses digital (computer controlled) copy systems with paper collating/sorting capability.

Printing selected pages of a PDF document
While viewing a PDF document with Adobe Reader (or Adobe Acrobat), any page or range of pages can be printed by a personal computer printer by:

Selecting the printer icon on the tool bar
Selecting **Print** from the Adobe **File** menu

Paper
The selection of the paper type to be used for printing a PDF document should be based on the target market for the document. For a user's manual with extensive graphics that is printed on both sides of a page the minimum recommended paper type is 24 pound. A heavier paper stock (26 – 30 pound) will reduce the 'bleed through' inherent with printed graphics. Typically the front and back cover pages are printed on heavy paper stock (50 to 60 pound).

Binding
Unlike the binding of a book or catalog, a user's manual distributed in as a PDF file will typically use 'comb' or 'coil' binding. This service is provided by most full service print shops. Coil binding is

suitable for documents with no more than 100 pieces of paper (24 pound). Comb binding is acceptable for documents with as many as 300 pieces of paper (24 pound). Most print shops stock a wide variety of 'combs'. The print shop can recommend the appropriate 'comb' based on the number of pages.

Pricing

The final cost for printing and binding a PDF document is based on:

- Quantity per print run
- Number of pages
- Paper type

The price range for printing and binding a PDF document similar to this user manual will be $15 to $30 (printed in Black & White) in quantities of 1 to 10 pieces.

Obtaining a Word 2000 version of this user manual

This user document was written using Microsoft's Word 2000. Qualified OEM's, Distributors, and Value Added Reps (VAR's) can obtain a copy of this document for

- Editing
- Customization
- Language translation.

Please contact Precision MicroControl to obtain a Word 2000 version of this document.

Index

PMC
Precision MicroControl Corp.

Precision MicroControl Corporation
2075-N Corte del Nogal
Carlsbad, CA 92009-1415 USA

Tel: (760) 930-0101
Fax: (760) 930-0222

www.pmccorp.com

Information: info@pmccorp.com
Technical Support: support@pmccorp.com