

# Compumotor

## Motion Toolbox™ User Guide

A Library of LabVIEW  
Virtual Instruments  
for Motion Control

*Developed by Snider Consultants, Inc.*

Compumotor Division  
Parker Hannifin Corporation  
p/n 88-013929-01 A



ABB Inc.

EXHIBIT 1015

# Motion Toolbox™ User Guide

A Library of LabVIEW® Virtual Instruments  
for  
Motion Control

Version 1.0

March 1994

Part Number 88-013929-01



SNIDER  
CONSULTANTS  
Inc.

© Copyright 1994 Snider Consultants, Inc.  
All Rights Reserved.

## Warnings

Because software controls machinery, test any software control for safety under all potential operating conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

## Limited Warranty

Compumotor warrants that the software will perform substantially in accordance with the accompanying material for a period of ninety (90) days from the date of receipt. Any implied warranties on the software are limited to ninety days. Some states do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

Compumotor's entire liability and your exclusive remedy shall be at Compumotor's option, either (a) return the price paid or (b) replacement of the software that does not meet Compumotor's Limited Warranty and that is returned to Compumotor with a copy of your receipt. This limited warranty is void if failure of the software has resulted from accident, abuse, or misapplication. Any replacement of software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

Compumotor disclaims all other warranties, either expressed or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the software and the accompanying written materials. This limited warranty gives you specific legal rights. You may have others which vary from state to state.

In no event shall Compumotor or its suppliers or distributors be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use or inability to use this Compumotor product, even if Compumotor has been advised of the possibility of such damages. Because some states do not allow the exclusion of limitation for liability for consequential or incidental damages, the above limitation may not apply to you.

### Copyright

Under copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or part, without the prior written consent of Snider Consultants, Inc.

### Trademarks

Motion Toolbox is a trademark of Snider Consultants, Inc.

Motion Architect is a registered trademark of Parker Hannifin Corporation, Compumotor Division.

LabVIEW is a registered trademark of National Instruments Corporation.

Windows is a registered trademark of Microsoft Corporation.

## Table of Contents

<b>Introduction</b> .....	<b>1</b>
Overview .....	1
Installation .....	1
Package Contents .....	1
Software Requirements .....	2
Pre-Installation Procedure .....	2
Installation Procedure .....	3
Motion Toolbox Registration .....	3
Getting Started .....	3
Related Publications .....	5
<b>Using Motion Toolbox</b> .....	<b>7</b>
Device Communication .....	7
Common VI Inputs and Outputs .....	7
Motion Toolbox Error Handling .....	9
6000 Controller Error Handling .....	9
Reducing VI load Time .....	9
Timer Resolution .....	10
VI Labeling Conventions .....	10
Bold and Non-bold .....	10
Default Inputs .....	10
<b>Counter &amp; Timer VIs</b> .....	<b>11</b>
Counter & Timer VI Descriptions .....	11
Start 6000 Timer .....	11
Stop 6000 Timer .....	12
Configure Encoder Input as Counter .....	12
Reset 6000 Hardware Counter .....	13
<b>Configuration VIs</b> .....	<b>15</b>
Configuration VI Descriptions .....	15
Set Motion Scaling Factors .....	15
Set Path Scaling Factors .....	16
Enable Scale Factors .....	17
Set Participating Axes .....	17
Configure Command Control .....	18
Set Continuous/Preset Mode .....	19
Set Absolute/Incremental Mode .....	19
Set Drive Resolution .....	20
Set Drive Fault Level .....	20
Enable Drive .....	21
Configure Feedrate Override .....	22
Enable Feedrate Override .....	23
Set Encoder Resolution .....	23
Set Encoder/Motor Step Mode .....	24

Configure Homing .....	24
Configure Position Maintenance.....	26
Enable Position Maintenance .....	27
Configure Stall Detection .....	27
Enable Stall Detection .....	28
Set Pulse Width .....	29
<b>Device Communication .....</b>	<b>31</b>
Device Communication VI Descriptions .....	31
Open AT6400 .....	31
Open 6200 .....	32
Close Device .....	33
Set Default Addr/Port .....	33
Reset 6000 .....	33
Set Error Action .....	34
Set Parameter Precision .....	34
Set AT6400 Polling Parameters .....	35
Set 6200 Polling Parameters .....	36
Send 6000 Block .....	36
Receive 6000 Block .....	37
Query 6000 .....	37
Download 6000 File .....	38
Enable Communications Tracing .....	38
Configure Communications Tracing .....	39
Delete Trace File .....	40
Command Snooper .....	40
<b>Fast Status VIs .....</b>	<b>41</b>
Fast Status VI Descriptions .....	41
Get Fast Status .....	41
Set Motor Pos Local Scaling .....	43
Set Encoder Pos Local Scaling .....	43
Set Velocity Local Scaling .....	44
Motor Position Parse .....	45
Encoder Position Parse .....	45
Velocity Parse .....	46
Pos. Captured Status Parse .....	47
Command Error Parse .....	47
Configuration Status Parse .....	48
Encoder Feedback Status Parse .....	48
Hard Limit Status Parse .....	49
Motion Status Parse .....	49
Soft Limit Status Parse .....	50
System Status Parse .....	51
Drive Status Parse .....	52
User Status Parse .....	53

Timer Status Parse.....	53
Analog Input Parse.....	54
Limit Input Status Parse.....	54
Joystick Status Parse.....	55
Digital Output Status Parse.....	56
Other Input Parse.....	56
Digital Input Parse.....	57
<b>I/O &amp; Limit VIs.....</b>	<b>59</b>
I/O & Limit VI Descriptions.....	59
Set 6000 Input Active Level.....	59
Enable 6000 Inputs.....	60
Set 6000 Input Function.....	61
Enable 6000 Input Functions.....	62
Set 6000 Input Debounce Time.....	62
Set 6000 Output Active Level.....	63
Enable 6000 Outputs.....	64
Set 6000 Output Function.....	64
Enable 6000 Output Functions.....	65
Set 6000 Output States.....	66
Configure Hard Limits.....	67
Enable Hard Limits.....	67
Configure Soft Limits.....	68
Enable Soft Limits.....	69
Enable Analog Input Override.....	69
Set Analog Input Override Voltage.....	70
Define 6000 User Status.....	71
Enable 6000 User Status.....	72
<b>Jogging &amp; Joystick VIs.....</b>	<b>73</b>
Jogging and Joystick VI Descriptions.....	73
Enable Jog Mode.....	73
Set Jog Velocity Low.....	74
Set Jog Velocity High.....	74
Set Jog Acceleration.....	75
Set Jog Deceleration.....	75
Enable Joystick Mode.....	76
Set Joystick Velocity Low.....	76
Set Joystick Velocity High.....	77
Set Joystick Acceleration.....	77
Set Joystick Deceleration.....	78
Setup Joystick Electronics.....	78
Set Joystick Analog Inputs.....	79
Set Joystick Zero.....	80

<b>Miscellaneous VIs .....</b>	<b>81</b>
Miscellaneous VI Descriptions .....	81
Numeric Event.....	81
Boolean Event .....	82
Boolean Transition .....	83
<b>Motion VIs .....</b>	<b>85</b>
Motion VI Descriptions .....	85
Initiate Motion.....	85
Stop Motion.....	86
Kill Motion .....	87
Set Velocity .....	88
Set Acceleration .....	88
Set Deceleration .....	89
Set Direction.....	89
Go Home.....	90
Initiate Linear Int. Motion.....	90
Set Position .....	91
Wait for Move Complete.....	92
Run Path .....	92
Set Path Vel & Acc.....	93
Run Program .....	93
<b>RP240 Display VIs .....</b>	<b>95</b>
RP240 VI Descriptions.....	95
Write Text to RP240 .....	95
Display Variable on RP240 .....	96
Set RP240 LED States.....	96
Clear RP240 Display .....	97
Position RP240 Cursor.....	97
Enable RP240 Jog Mode.....	98
Set RP240 Password.....	99
<b>Variable &amp; Transfer VIs.....</b>	<b>101</b>
Variable & Transfer VI Descriptions .....	101
Set Numeric Variable.....	101
Set Binary Variable.....	102
Set Binary Variable by Byte.....	102
Set String Variable.....	103
Transfer Numeric Variable.....	103
Transfer Binary Variable.....	104
Transfer String Variable .....	104
Transfer Captured Positions .....	105
Miscellaneous Transfers.....	106



<b>Appendix .....</b>	<b>107</b>
Motion Toolbox Error Codes .....	107
Technical Support .....	108
Compumotor Bulletin Board Service .....	108
Systems Integration and Consulting .....	108
<b>Motion Toolbox VI Index (alphabetical) .....</b>	<b>109</b>

# Introduction

---

## Overview

Motion Toolbox is a library of LabVIEW VIs for Compumotor's 6000 series of motion controllers. Motion Toolbox allows LabVIEW programmers to develop motion control systems for a wide range of applications including automated test and manufacturing, medical and biotech, metering and dispensing, machine control, and laboratory automation. Motion Toolbox supports both the Compumotor AT6400 4-Axis and 6200 2-Axis Indexers.

Motion Toolbox provides developers with the following capabilities and more:

- Motion control including velocity, acceleration, deceleration, go, stop, kill, etc.
- Setup, control, and command file transfer
- Counter and timer configuration and control
- Indexer, encoder, and drive configuration
- Home, hardware limit, and soft limit configuration
- Jogging and joystick configuration
- I/O setup and function configuration
- Fast status querying of I/O, limit, home, motor and encoder position, velocity, etc.

To use Motion Toolbox effectively, you should have a working knowledge of LabVIEW and have at least worked through the LabVIEW Tutorial manual.

## Installation

### Package Contents

- *Motion Toolbox* installation disks
- *Motion Architect™* installation disks
- *Motion Toolbox User Guide*
- User registration card
- License agreement

*Motion Toolbox User Guide*

If any of these items are missing, please contact your local supplier or Compumotor's Customer Service Department at 800-358-9068 or 707-584-7558.

## Software Requirements

- LabVIEW for Windows version 3.0. See the *Required System Configuration* in the *LabVIEW for Windows Release Notes*.
- *Motion Architect* version 2.2 or higher.
- If you are using an AT6400 Controller, you need software revision 2.2 or higher of the AT6400 operating system.
- If you are using a 6200 Controller, you need software revision 2.1 or higher of the 6200 system software.

## Pre-Installation Procedure

1. Install your 6000 Series controller as indicated in the Compumotor User Guide for your product.
2. Install *Motion Architect* 2.2, if it is not already on your computer.
3. Using *Motion Architect*, ensure the 6000 controller functions properly. For help, refer to the *Motion Architect* documentation shipped with your controller.
4. Copy the following files from your *Motion Architect* 2.2 (or higher) directory to your LabVIEW root directory.
  - WIN6400.DLL
  - METER.DLL
  - AT6400.OPS

**Warning:** *Motion Toolbox* cannot communicate with the AT6400 controller without these files in the LabVIEW root directory.

**Warning:** *It is important that the same WIN6400.DLL, METER.DLL, and AT6400.OPS files exist in both the Motion Architect directory and LabVIEW root directory. If the files are not the same, conflicts may arise when using Motion Toolbox in the same Windows session as Motion Architect. You should always use the latest version of these files that are available.*

## Installation Procedure

*Motion Toolbox* is distributed in compressed form on 3.5" floppy disks. A Windows setup program is provided to install the software on your hard drive. To install *Motion Toolbox*, complete the following steps:

1. Start Windows.
2. Place disk 1 of the *Motion Toolbox* installation disks in your 3.5" floppy drive.
3. From the Program Manager, choose File/Run.
4. Type A:SETUP.EXE and select OK. If drive A is not your 3.5" drive, substitute A: with the appropriate drive letter.
5. Follow the directions given by the setup program for the remainder of the installation.

## Motion Toolbox Registration

To receive free technical support and update information, please fill out the postage paid software registration form and drop it in the mail.

## Getting Started

The following steps will help you get started using *Motion Toolbox*.

1. If you are not already familiar with LabVIEW, you should at least work through the LabVIEW tutorial manual.
2. Read the *Using Motion Toolbox* chapter of this user guide.
3. Become familiar with the *Motion Architect* software. The *Motion Architect* terminal program is an excellent utility for debugging 6000 series controller applications. If you encounter problems using *Motion Toolbox*, use *Motion Architect* to verify your 6000 controller is functioning properly. See the description for the *Download 6000 File VI* for more ways to use *Motion Architect* for application development.

*Motion Toolbox User Guide*

4. Read the indexer user guide for your product. If you wish, you can skim the examples on programming the 6000 controller as *Motion Toolbox* will isolate you from most of these details.
5. Read the *6000 Series Command Language Overview* chapter in the *6000 Series Software Reference Guide*. This will provide important information on system performance, 6000 inputs and outputs, and programming restrictions. Because *Motion Toolbox* essentially assembles 6000 series commands and sends them to the controller, you may want to read the commands related to your application in the *6000 Series Software Reference Guide*. Each VI description in this manual lists the applicable 6000 commands related to the VI.
6. Review the example and demonstration VIs included with *Motion Toolbox*. Study the diagrams to see how *Motion Toolbox* VIs are used. Try modifying and adding your own ideas to these examples to get a feel for programming with *Motion Toolbox*. Make sure to rename the example and demonstration VIs before modifying.

If you accepted the default demo directory when installing *Motion Toolbox*, you will find the example and demonstration VIs located in MTDEMO off the LabVIEW root directory.

- a. Open the *Simple Preset Motion.VI* located in the \LABVIEW\MTDEMO\EXAMPLES.LLB VI Library.
- b. If you have an AT6400, verify that the address setting in the VI diagram is the same as your installed AT6400. If you have a 6200, you must replace the *Open AT6400 VI* with the *Open 6200 VI* and change the input address from 768 to the proper serial communications port (usually COM1 or COM2). Put the VI in Edit Mode (CTRL-M) to make these changes.
- c. Return to the front panel, the VI is now ready to run. Press the Run button (arrow button).
- d. The VI is now running. Enter a distance to travel and press the GO button on the front panel to initiate motion. The commanded distance is relative to an absolute counter in this case. In order for the motor to move, the value displayed in the distance control must differ from the current position indicated on the chart.
- e. If the motor does not move, consult the *Getting Started* Section of the Hardware Reference guide for your particular controller.

7. You are now ready to begin work on your application. Be sure to read through the reference descriptions of the *Motion Toolbox* VIs to see what is available.

## **Related Publications**

- *AT6400 Indexer User Guide*
- *6200 Indexer User Guide*
- *6000 Series Software Reference Guide*
- *Motion Architect User Guide*
- *LabVIEW for Windows 3.0* documentation set
- *LabVIEW Installation Notes*
- *Current Parker Compumotor Motion Control Catalog*

# Using Motion Toolbox

## Device Communication

Motion Toolbox supports both the PC bus-based AT6400 and serial communications-based 6200. To communicate with the AT6400, Motion Toolbox calls low level DLL routines through a LabVIEW Code Interface Node (CIN). To communicate with the 6200, Motion Toolbox uses LabVIEW's serial communication VIs. From a programmer's standpoint using Motion Toolbox, the differences in developing applications for the two products are minor.

All VIs with an *Addr/port* input indicate the VI is applicable to both 6000 products and will automatically direct communications appropriately. A VI with its *Addr/port* value less than, or equal to, 16 automatically directs communication to the 6200 while an *Addr/port* value above 16 directs communication to the AT6400. Before attempting to communicate with either product, you must establish device communications using *Open 6200 VI* or *Open AT6400 VI* appropriately. When using 6200 controllers, you are limited to a single controller per communications port. You cannot daisy-chain 6200 controllers when using Motion Toolbox. To control more than two 6200 controllers, use a multi-port serial card or an external multi-port device that is compatible with LabVIEW.

Motion Toolbox consists of three types of VIs. Command oriented (*Initiate Motion, Kill, Start Timer, etc.*), transfer oriented (*Transfer Numeric Variable*), and fast status oriented (*Get Fast Status, Motion Status Parse, etc.*). Command oriented VIs essentially generate 6000 Series command strings and send them to the 6000 controller. Transfer or query oriented VIs send a prompt string to the 6000 controller and return the response. For Fast Status VIs, Motion Toolbox uses a binary Fast Status transfer from the 6000 controller and converts the blocks of information to LabVIEW clusters. This is accomplished by the *Get Fast Status VI*. Additional Fast Status VIs parse the output clusters of *Get Fast Status* for convenient programming.

## Common VI Inputs and Outputs

Most Motion Toolbox VIs have several common input and output terminals. They are defined below to avoid redundancy in VI descriptions.

**TF** **Execute VI?** Specifies whether the VI should execute normally or do nothing and return.

In applications dealing with control cards, data acquisition cards, or instruments, should send setup and configuration information to the card only when the information changes. This avoids needless communication that degrades the performance of the application and the device.

A solution is to embed a VI that communicates with the device within a true/false case statement that executes only when appropriate. In *Motion Toolbox*, the burden of the case statement is moved to the communication VI itself. The *Execute VI?* input is essentially the input to this case statement.

True: Execute VI normally.

False: Do nothing and return.

**TF** **Immediate?** Dictates whether the command(s) assembled by the VI are sent to the 6000 controller as immediate or non-immediate commands. Immediate 6000 commands are pre-pended by the '!' specifier. Immediate commands are executed as soon as the controller receives them. Non-immediate commands are buffered and will execute when the previous non-immediate command(s) are completed.

True: Pre-pend commands by the immediate specifier.

False: Do not pre-pend commands by the immediate specifier.

**IS2** **Addr/port** specifies either the switch-selected address or the AT6400 product or the serial port connected to a 6200 product.

6200 Product:

- 1: Communications port 1
- 2: Communications port 2
- 3: Communications port 3
- 4: Communications port 4

AT6400 Product:

Valid AT6400 address (768 to 1023 decimal, 300 to 3FF hex)

**IS2** **Axis.** Specifies the axis the VI affects. Valid 6200 axes are 1-2. Valid AT6400 axes are 1-4.



**132** Error returns any error conditions from the VI. Possible error codes are given as an appendix to this user guide.

## Motion Toolbox Error Handling

All Motion Toolbox VIs capable of generating an error have an *Error* output terminal. The possible error codes are given as an appendix to this user guide.

All Motion Toolbox VIs that have the *Error* output automatically call an error handling VI upon encountering a non-zero *Error* value. Using the *Set Error Action* VI, you can dictate the action taken at the time of an error occurrence. This VI allows you to disable error enunciation altogether or to display a message describing the nature of the error. See the *Set Error Action* VI description for more information. Depending on the nature of the motion control application, you may want to augment the capabilities of the *Motion Toolbox Error Handler* by adding additional error actions.

## 6000 Controller Error Handling

If an invalid command such as a range or syntax error is encountered, the 6000 controller sets bit 11 of the system status register to indicate a command error has occurred. You can check the status of this bit using the *Command Error Parse* VI. You may also retrieve the command that generated the error using the *Miscellaneous Transfers* VI. It is good practice to include a check for the command error in your program during development. If desired, you could disable the check after your program is running and debugged.

## Reducing VI load Time

The first time a VI from Motion Toolbox is loaded, it may need to be re-compiled. This will happen if you do not use the default directory when loading Motion Toolbox. LabVIEW compiling adds overhead to the loading process that increases the VI's load time. To avoid the compilation process in subsequent use of the VI, save it before closing. Make sure not to modify the VI in any way!

A better way is to compile all the Motion Toolbox .LLB files using LabVIEW's mass compile feature. To do this, select *Mass Compile...* under LabVIEW's *File* menu. Now select the \MOTION directory (should be located off LabVIEW's VI.LIB directory) that holds the Motion Toolbox VIs. You can also select each .LLB file separately. If you elect to mass compile the entire MOTION directory of .LLBs, it may take a while to execute. See the *LabVIEW 3.0 for Windows User Manual* for more information on mass compiling.

## Timer Resolution

The default resolution of the timer functions included with LabVIEW for Windows is 55 milliseconds. You can increase the resolution to 1 ms by altering the LabVIEW preferences file (see LabVIEW release notes). Using the 1 ms timer resolution dramatically increases the performance of 6000 communications, particularly with the 6200. Under certain circumstances, however, the increased interrupt load caused by the higher resolution timer may exceed your PC's capacity to handle the interrupts. Please read the section *Timer Resolution* in the LabVIEW release notes to see if a 1 ms resolution will be appropriate for your application.

## VI Labeling Conventions

### Bold and Non-bold

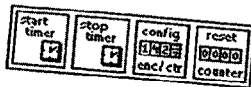
The I/O parameters of Motion Toolbox VIs are labeled in bold and non-bold text. A bold label indicates the parameter is important and is normally wired when the VI is used in an application. Non-bold labels indicate that the parameter can often be left unwired, thereby using the default, without affecting the VIs primary function.

### Default Inputs

Parenthesis ( ) on the label of an input parameter indicate the default value the VI will automatically use if the terminal is left unwired. The inputs of many VIs in Motion Toolbox default to "no change" meaning that no change is made to the current setting.

## Counter & Timer VIs

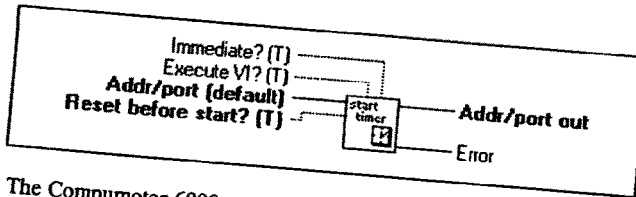
This chapter contains descriptions of the counter and timer VIs included with Motion Toolbox. The figure below displays the *Counter & Timer* function palette.



Counter & Timer Function Palette

### Counter & Timer VI Descriptions

#### Start 6000 Timer



The Compumotor 6000 controller has a two-millisecond resolution timer built into the product. The timer can be started, stopped and reset without interrupting any motion that is taking place. This VI starts the 6000 timer. To get the value of the timer, use the *Get Fast Status* and *Timer Status Parse* VIs.

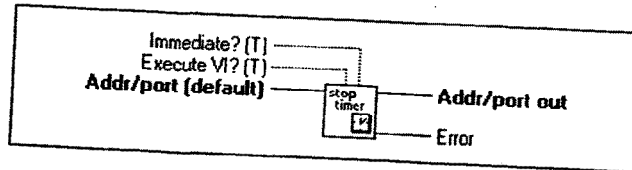
**TF** **Reset before start?** A true resets the 6000 timer before starting.

True: Reset and start timer.

False: Resume counting from current value.

6000 command reference: TIMST

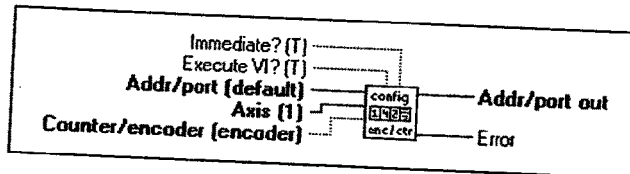
## Stop 6000 Timer



Stops the 6000 timer. To get the value of the timer, use the *Get Fast Status* and *Timer Status Parse* VIs.

6000 command reference: TIMSTP

## Configure Encoder Input as Counter



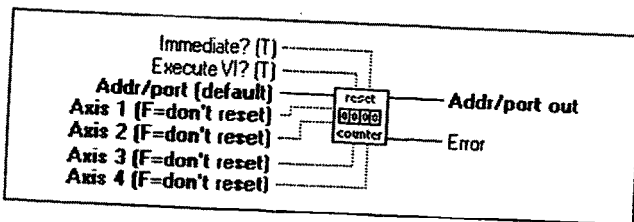
Configures the encoder input of the specified *Axis* to be used as a counter. The hardware counter can either count up or count down. The direction of the count is specified by the signal on the encoder channel B+ and B- connections. The default condition for the controller uses the inputs for encoders. Only use this VI if you want to change the inputs to a counter.

**TF** Counter/encoder.

True: Configure input as up/down counter.  
False: Configure input for encoder feedback.

6000 command reference: CNTE

## Reset 6000 Hardware Counter



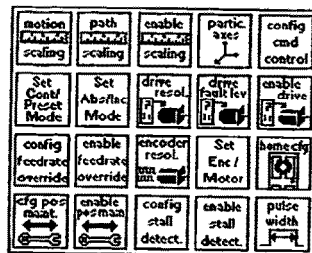
Clears specified encoder registers that are configured as counters. The counters can also be cleared by applying a positive differential signal to Z+ and Z-.

**YF** Axis 1-4. A true resets the respective counter.

6000 command reference: CNTR

## Configuration VIs

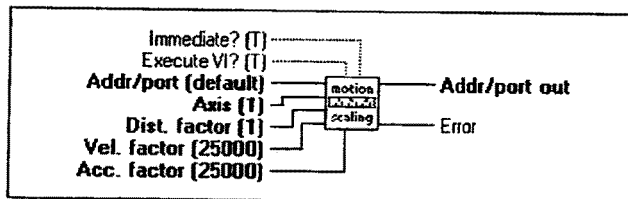
This chapter contains descriptions of the configuration VIs included with Motion Toolbox. The figure below displays the *Configuration* function palette.



Configuration Function Palette

## Configuration VI Descriptions

### Set Motion Scaling Factors



Sets scale factors for distance, velocity and acceleration. The 6000 controller internally multiplies the values issued via the *Set Distance*, *Set Velocity* and *Set Acceleration* VIs by the appropriate scaling factor provided scaling is enabled by the *Enable Scale Factors* VI.

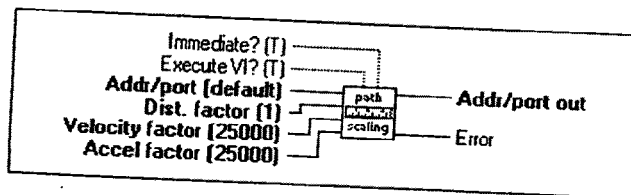
**152** **Dist. factor** specifies the distance scale factor for the respective axis. Valid range is 1 to 999,999.

**IS2** Velocity factor specifies the velocity scale factor for the respective axis. Valid range is 1 to 999,999.

**IS2** Accel factor specifies the acceleration scale factor for the respective axis. Valid range is 1 to 999,999.

6000 command reference: SCLD, SCLV, SCLA

### Set Path Scaling Factors



Sets path scale factors for distance, velocity and acceleration. If scaling is enabled (*Enable Scale Factors VI*), the 6000 controller internally multiplies the distance, velocity, and acceleration values used in path contouring and linear interpolated moves by the respective scale factor.

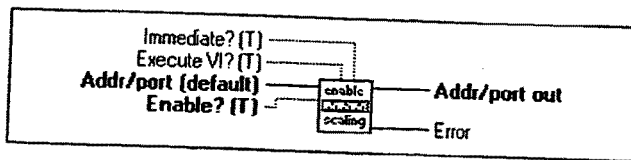
**IS2** Dist. factor specifies the path distance scale factor for contouring and linear interpolated moves. Valid range is 1 to 999,999.

**IS2** Velocity factor specifies the path velocity scale factor for contouring and linear interpolated moves. Valid range is 1 to 999,999.

**IS2** Accel factor specifies the path acceleration scale factor for contouring and linear interpolated moves. Valid range is 1 to 999,999.

6000 command reference: PSCLD, PSCLV, PSCLA

### Enable Scale Factors



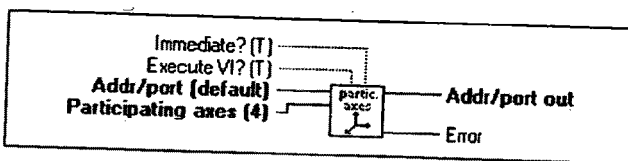
Enables and disables the acceleration, distance and velocity scaling factors. When enabled, all distance, velocity, and acceleration values are multiplied by the appropriate scale factor.

Enable?

True: Enable scale factors.  
False: Disable scale factors.

6000 command reference: SCALE

### Set Participating Axes



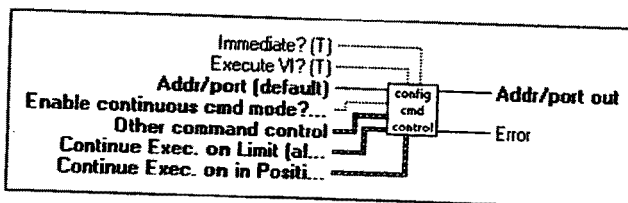
Defines the total number of axes to be controlled by the 6000 series product. The 6000 controller defaults to control all axes for the product.

Participating axes sets the number of axes to be controlled by the 6000 controller. Valid range is zero to four and is product dependent.

6000 command reference: INDAX



## Configure Command Control



Configures several run-time modes in the 6000 controller. In many applications it is appropriate to execute this VI with the default settings. You may want to experiment with different settings for your application. Refer to the *6000 Series Software Reference Guide* for a full explanation of these configuration modes.

**TF** **Enable continuous cmd mode?** Normally, when a motion command is received, command processing is temporarily paused until motion is complete. When continuous command mode is enabled, command processing continues while motion is in progress.

True: Enable continuous command mode.

False: Disable continuous command mode.

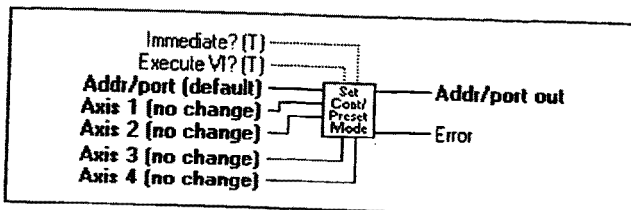
**Str** **Other command control** determines how the controller will respond after receiving a KILL, PAUSE/RESUME or STOP command.

**POS** **Continue Exec. on Limit** determines whether the controller's command buffer will be saved upon hitting an end-of-travel limit or a soft limit. If enabled, all commands following the command currently being executed will remain in the buffer when a limit is encountered. Otherwise the commands in the buffer will be discarded.

**POS** **Continue Exec. on In Position** determines whether the command processing will pause until the in-position signal is received from the drive. If enabled, command processing is paused until the in-position input is active.

**6000 command reference:** COMEXC, COMEXK, COMEXL, COMEXP, COMEXR, COMEXS

### Set Continuous/Preset Mode



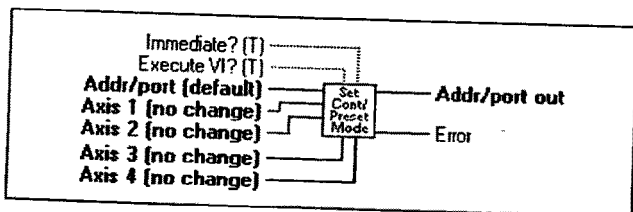
Specifies whether the axes are to move in a continuous fashion at a specified velocity or move a preset distance established with the *Set Distance VI*.

**IS2** Axis 1-4 determines the mode for the respective axis.

- 0: Configure axis for preset mode.
- 1: Configure axis for continuous mode.
- 2: Do not change active setting.

6000 command reference: MC

### Set Absolute/Incremental Mode



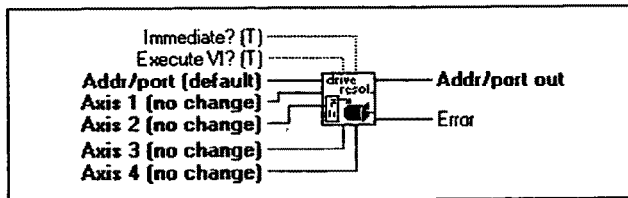
Specifies whether subsequent moves are made with respect to the current position (incremental) or with respect to an absolute zero position.

**IS2** Axis 1-4 determines the mode for the respective axis.

- 0: Configure axis for incremental mode.
- 1: Configure axis for absolute mode.
- 2: Do not change active setting.

6000 command reference: MA

### Set Drive Resolution

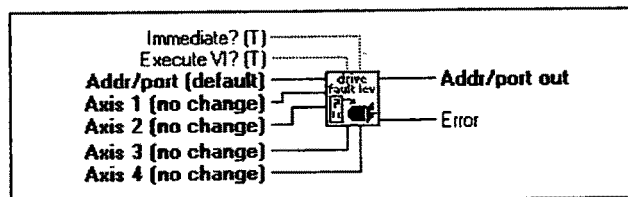


Configures the step motor controller resolution to match that of the motor/drive to which it is attached. This is necessary to accurately calculate motor velocities and accelerations. The default resolution is 25,000 steps per revolution for Compumotor step motor controllers.

**IS2** Axis 1-4 determines the resolution for the respective axis. Valid range is 200 to 1,024,000 steps per revolution.

6000 command reference: DRES

### Set Drive Fault Level



Sets the drive fault level for each axis. To enable the drive fault input, you must use the *Enable 6000 Input Functions VI* to enable input functions. Use the following table for setting the drive fault level for Compumotor drives.

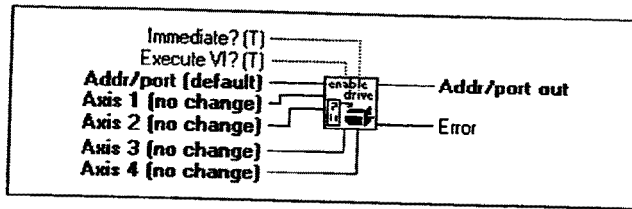
Compumotor Product	Drive Fault Level
BL,L,LE,PS7,UD2,UD5 & UD12	Active Low
Compumotor Plus, LN, OEMSeries, S & Z	Active High

**IS2** Axis 1-4 determines the drive fault level for the respective axis.

- 0: Active Low.
- 1: Active High.
- 2: Do not change active setting.

6000 command reference: DRFLVL

### Enable Drive



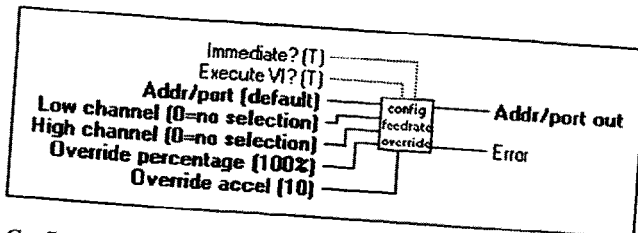
Energizes or de-energizes a Compumotor motor/drive combination.

**IS2** Axis 1-4 determines the status of the respective axis.

- 0: Disable drive.
- 1: Enable drive.
- 2: Do not change current setting.

6000 command reference: DRIVE

## Configure Feedrate Override

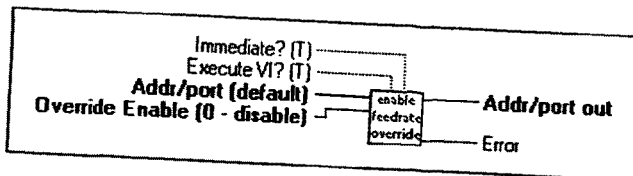


Configures parameters for the feedrate override feature. Feedrate override is used to synchronously scale all phases of motion (except distance) on all axes. The amount of scaling is expressed in terms of percentage from 0 to 100. The percentage of feedrate can be controlled by an analog voltage or by the *Override percentage* input of this VI. To enable feedrate override, use the *Enable Feedrate Override VI*.

- IS2** **Low channel** defines the analog channel that will be used when the axis select input (pin 15 of joystick connector) is low.
- IS2** **High Channel** defines the analog channel that will be used when the axis select input (pin 15 of joystick connector) is high.
- IS2** **Override percentage** specifies the percentage by which motion velocity will be scaled when feedrate override is enabled. Valid range is 0 to 100 percent.
- IS2** **Override accel** specifies the acceleration and deceleration to use when the velocity is changing due to a change in one of the analog inputs, or when the software feedrate override percentage is changing. Valid range is 1 to 50,000 percent/sec<sup>2</sup>.

6000 command reference: FRA, FRH, FRL, FRPER

## Enable Feedrate Override



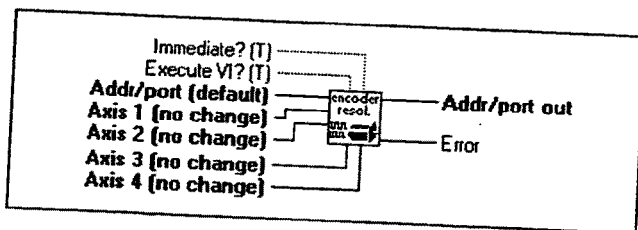
Enables or disables feedrate override on all axes. When using the hardware override, the analog channel defined with the *Configure Feedrate Override VI* is used to scale the motion velocity. At 0 VDC the percentage is 0% and at 2.5 VDC is 100%. To have programmatic control of the feedrate percentage, use the software feedrate option of this VI and the *Overrate percentage* input of the *Configure Feedrate Override VI*.

### IS2 Override Enable

- 0: Disable feedrate override.
- 1: Enable hardware feedrate override.
- 2: Enable software feedrate override.

6000 command reference: FR

## Set Encoder Resolution

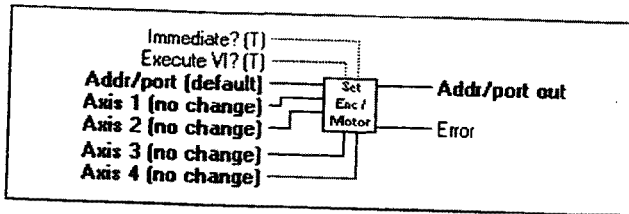


Configures the encoder counts per revolution (post quadrature) for the specified axis. This setting allows the controller to correctly interpret move distances, velocities, and accelerations while in the encoder step mode.

- IS2 Axis 1-4 determines the encoder resolution for the respective axis. Valid range is 200 to 65,535 counts per revolution.

6000 command reference: ERES

### Set Encoder/Motor Step Mode



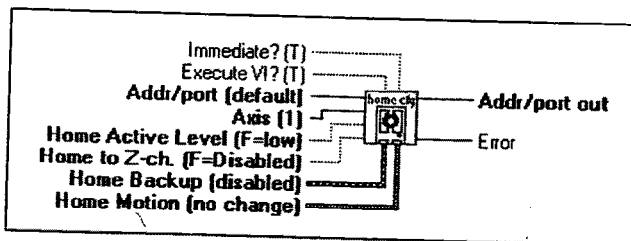
Specifies whether the move distances are based on motor steps or encoder steps.

**132** Axis 1-4 determines step mode for the respective axis.

- 0: Configure axis for motor step mode.
- 1: Configure axis for encoder step mode.
- 2: Do not change current setting.

6000 command reference: ENC

### Configure Homing



Configures various settings for the specified axis used when homing.

**TF** **Home Active Level** defines the active state of the home input.

True: Active level is high.  
False: Active level is low.

**TF** **Home to Z-ch.** enables homing to an encoder Z-channel after the initial home input has gone active.

True: Enable home to Z-channel.  
False: Disable home to Z-channel.

**50a** **Home Backup** is a cluster of controls that dictates home backup configuration. The cluster contains the following variables:

**Backup Enable.** Enables or disables the home backup facility.

True: Enable home backup.  
False: Disable home backup.

**Final Direction** specifies the direction of travel a motor will take upon its final approach of the homing sequence.

True: Set final direction to counter-clockwise.  
False: Set final direction to clockwise.

**Final Velocity** specifies the velocity to use when the home algorithm does its final approach. Valid range is 0 to 1,600,000 steps per second.

**Ref. Edge** specifies which edge of the home switch the homing operation will consider as its final destination.

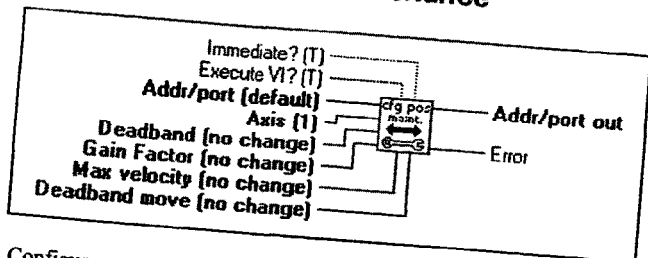
True: Define counter-clockwise edge as final destination.  
False: Define clockwise edge as final destination.

**50b** **Home Motion** is a cluster of double-precision values that specify the velocity, acceleration and deceleration, respectively, to use when the home algorithm begins its initial go-home move.

6000 command reference: HOMA, HOMAD, HOMBAC, HOMDF, HOMEDG, HOMLVL, HOMV, HOMVF, HOMZ



## Configure Position Maintenance



Configures position maintenance parameters for the specified axis. Position maintenance allows the step motor controller to *servo* (adjust) the motor until the correct encoder position is achieved. This occurs at the end of the move (if the final position is incorrect) or any time the controller senses a change in position while the motor is at zero velocity. To enable the position maintenance mode, use the *Enable Position Maintenance VI*.

**IS2** **Deadband** establishes the maximum encoder count error that is allowed at the end of a move to be considered in position when position maintenance is enabled. Valid range is 0 to 99,999,999 encoder counts.

**IS2** **Gain Factor** establishes the error correction velocity for the position maintenance move. The correction velocity is this value times the position error. Valid range is 0 to 999,999.

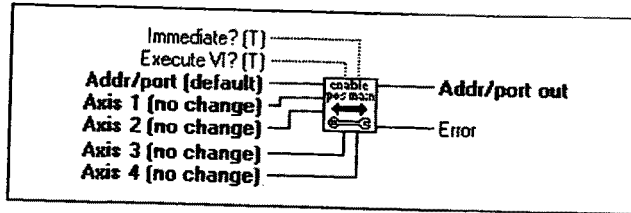
**DBL** **Max velocity** establishes the maximum velocity for any position maintenance move. Valid range is 0 to 1,600,000 units/sec.

**IS2** **Deadband move** configures a mode to determine if command processing should pause until the specified axis is within the allowable error as defined with the deadband input of this VI.

- 0: Do not pause command processing until in position.
- 1: Pause command processing until in position.
- 2: Do not change active setting.

6000 command reference: EMOVDB, EPMDB, EPMG, EPMV

## Enable Position Maintenance



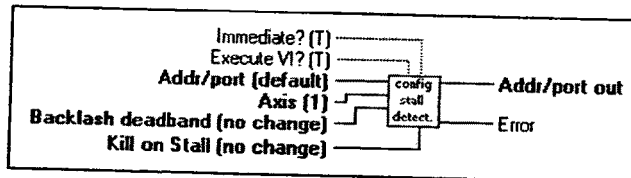
Enables and disables the position maintenance mode for the respective axis. This mode is only active while in encoder step mode. Setup configurations for each axis are determined with the *Configure Position Maintenance VI*.

**I32** Axis 1-4 determines the mode for the respective axis.

- 0: Disable position maintenance mode.
- 1: Enable position maintenance mode.
- 2: Do not change current setting.

6000 command reference: EPM

## Configure Stall Detection



Configures stall detection on the respective axis. To enable these settings, use the *Enable Stall Detection VI*.

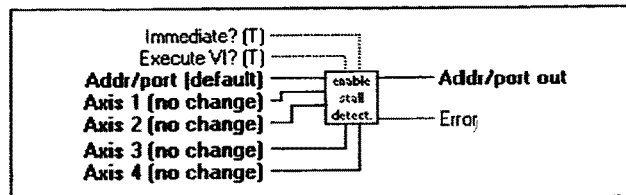
**I32** Backlash deadband establishes the maximum number of motor steps that a move can fall behind after a change in direction before a stall detection is initiated. Valid range is 0 to 99,999,999 motor steps.

**IS2** Kill on Stall indicates if pulses to the drive are immediately stopped when a stall has been detected on the specified axis. Stall detection must be enabled before this command will have any effect.

- 0: Disable Kill on Stall.
- 1: Enable Kill on Stall.
- 2: Do not change current setting.

6000 command reference: ESDB, ESK

### Enable Stall Detection

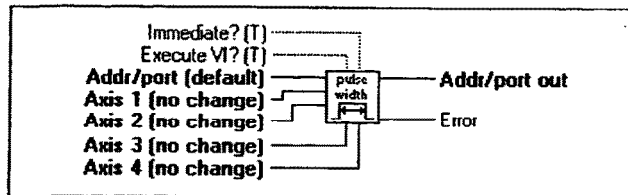


Determines if stall conditions defined with the *Configure Stall Detection VI* are monitored.

- IS2** Axis 1-4 determines the mode for the respective axis.
- 0: Disable stall detection mode.
  - 1: Enable stall detection mode.
  - 2: Do not change current setting.

6000 command reference: ESTALL

## Set Pulse Width



Sets the step output width for the corresponding axis. The pulse width is described as the time the pulse is active, or on. When the pulse width changes from the default 0.3 $\mu$ s, the maximum velocity range is reduced.

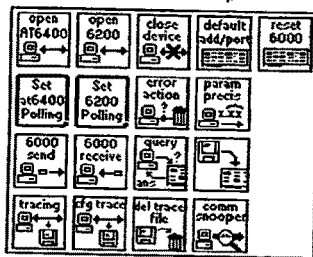
**IS2** Axis 1-4 determines the pulse width for the respective axis.

- 0: Do not change current setting.)
- 1: 0.3  $\mu$ s pulse width.
- 2: 0.5  $\mu$ s pulse width.
- 3: 1.0  $\mu$ s pulse width.
- 4: 2.0  $\mu$ s pulse width.
- 5: 5.0  $\mu$ s pulse width.
- 6: 10.0  $\mu$ s pulse width.
- 7: 16.0  $\mu$ s pulse width.
- 8: 20.0  $\mu$ s pulse width.

6000 command reference: PULSE

## Device Communication

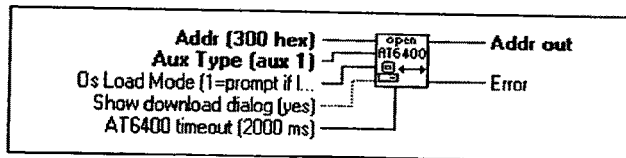
This chapter contains descriptions of the device communication VIs included with Motion Toolbox. The figure below displays the *Device Communication* function palette.



Device Communication Function Palette

### Device Communication VI Descriptions

#### Open AT6400



Opens AT6400 at the specified address, downloads the AT6400 operating system, and initializes the card for Motion Toolbox communication. If *delete trace file upon open* (see *Configure Communications Tracing VI*) is enabled, the trace file is deleted.

**152** Aux Type specifies the auxiliary board use.

- 0: AUX1
- 1: AUX2

**IS2** OS Load Mode.

- 0: Always load operating system.
- 1: Prompt user if operating system is already loaded.

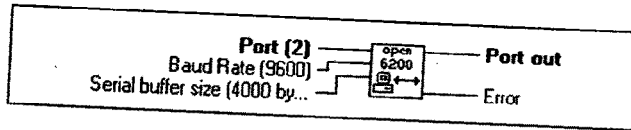
**TF** Show download dialog.

- True: Display meter dialog indicating progress of downloading AT6400 operating system.
- False: Do not display dialog while downloading operating system.

**IS2** AT6400 Timeout. Sets the AT6400 board response timeout. This value is used by the driver DLL when communicating with the AT6400 controller and is not related to the AT6400 polling parameters.

6000 command reference: N/A

## Open 6200



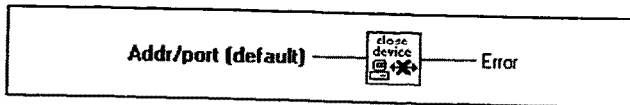
Opens 6200 at specified serial port and initializes the unit for Motion Toolbox communication. If *delete trace file upon open* (see *Configure Communications Tracing VI*) is enabled, the trace file is deleted.

**U16** Baud Rate. Specifies the serial communications baud rate. The 6200 controller normally uses the default 9600 baud rate.

**U16** Serial buffer size. Indicates the size of the input and output buffers allocated in PC memory for communications through the specified port. If the buffer size is less than or equal to 1 K, then 1 K is used as the buffer size.

6000 command reference: N/A

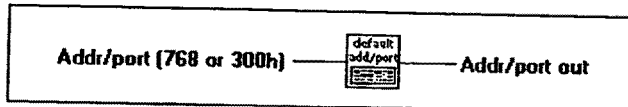
### Close Device



Closes the communications link with the 6000 product. If the device is a 62XX product, the serial port is released.

6000 command reference: N/A

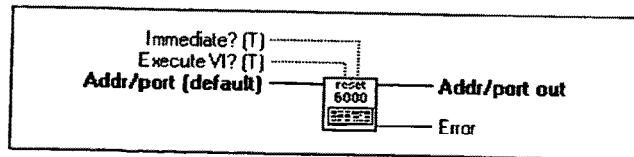
### Set Default Addr/Port



Sets the value of the default address/port. This forces all subsequent calls to Motion Toolbox VIs to use this value if the *Addr/port* terminal is not wired.

6000 command reference: N/A

### Reset 6000

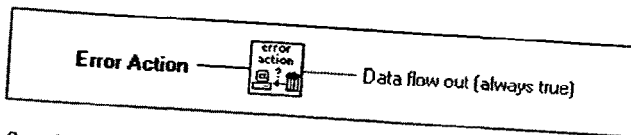


Resets the 6000 controller and re-initializes it to a state appropriate for Motion Toolbox communications.

**Note:** This action deletes all volatile information in the product.

6000 command reference: RESET

## Set Error Action



Sets the action to execute upon Motion Toolbox encountering an error.

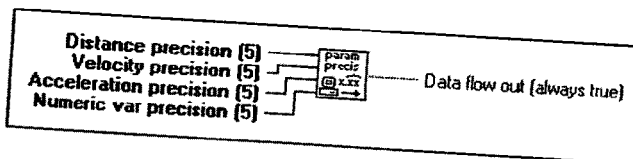
**I32** Error Action.

- 0: Do nothing.
- 1: Display ok/Stop dialog box describing the error.

**TF** Data flow out is always true. It can be used to force correct sequencing of this VI.

6000 command reference: N/A

## Set Parameter Precision



Sets the variable precision used when sending commands to the 6000 controller. Reducing the precision decreases the number of characters to transmit to the controller thereby improving communications performance. When building an application, set the precisions to the maximum value necessary for the corresponding parameter type. Valid range for all precision inputs is 0 to 5.

**I32** Distance precision specifies the number of decimal places to use when sending distance values to the 6000 controller.

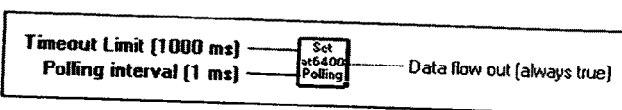
**I32** Velocity precision specifies the number of decimal places to use when sending velocity values to the 6000 controller.



- IS2** Acceleration precision specifies the number of decimal places to use when sending acceleration values to the 6000 controller.
- IS2** Numeric var Precision specifies the number of decimal places to use when sending variable values to the 6000 controller.
- TF** Data flow out is always true. It can be used to force correct sequencing of this VI.

6000 command reference: N/A

### Set AT6400 Polling Parameters

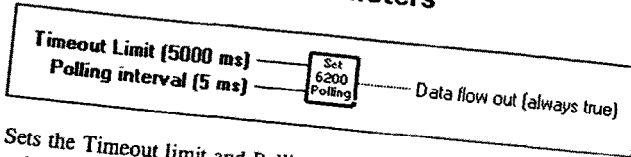


Sets the *Timeout limit* and *Polling interval* for AT6400 communications. These values are used when executing AT6400 queries and transfers.

- IS2** **Timeout Limit** is the amount of time to wait in milliseconds for a response from the 6000 controller when executing commands that elicit a response (e.g., query oriented commands).
- US2** **Polling Interval** specifies the amount of time in milliseconds between checking for a response during query and transfer operations.
- TF** **Data flow out** is always true. It can be used to force correct sequencing of this VI.

6000 command reference: N/A

## Set 6200 Polling Parameters



Sets the Timeout limit and Polling interval for 6200 communications. These values are used when executing 6200 queries and transfers.

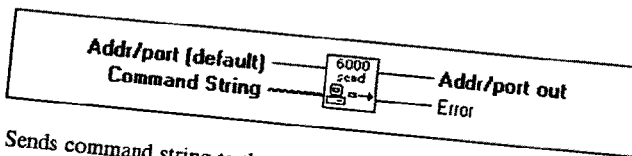
**I32** **Timeout Limit** is the amount of time to wait in milliseconds for a response from the 6000 controller when executing commands that elicit a response (e.g., query oriented commands).

**I32** **Polling Interval** specifies the amount of time in milliseconds between checking for a response during query and transfer operations.

**TF** **Data flow out** is always true. It can be used to force correct sequencing of this VI.

6000 command reference: N/A

## Send 6000 Block

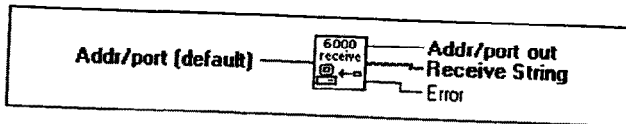


Sends command string to the 6000 controller. The command string is a concatenation of valid 6000 commands. To execute an immediate command, prepend the command with an immediate specifier ("!").

**abc** **Command string** is the command text to send the 6000 controller.

6000 command reference: N/A

## Receive 6000 Block

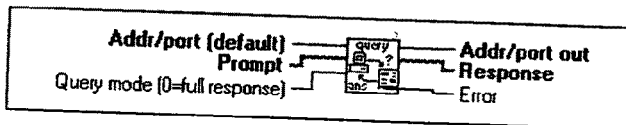


Receives string from the 6000 controller.

**abc** Receive string is the response text from the 6000 controller.

6000 command reference: N/A

## Query 6000



Sends a single prompt string to the 6000 controller and returns the response. The polling scheme used when querying is dictated by either the *Set AT6400 Polling Parameters VI* or *Set 6200 Polling Parameters VI* appropriately.

**Note:** Prompts should not include an immediate specifier ("!") as it is added automatically. Nor should prompts include a command delimiter (e.g., ":") as it is added automatically.

**abc** **Prompt** contains the command string sent to the 6000 controller to elicit a response.

**132** **Query mode.** Determines how response string is parsed.

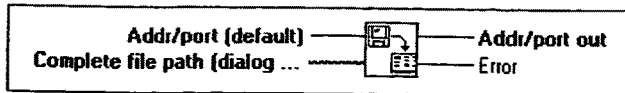
0: Return full response.

1: Return all characters between prompt and EOT.

**abc** **Response** is the response text from the 6000 controller.


6000 command reference: Transfer Commands

## Download 6000 File



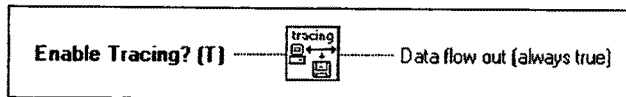
Downloads the contents of the specified file to the 6000. This VI opens the specified file beforehand and closes it afterwards. The file must contain valid 6000 commands. The file can be a list of 6000 series commands or a program or path definition.

Use *Motion Architect* to develop 6000 series program and path files and download them using Motion Toolbox. *Motion Architect's* setup facility is a great way to generate setup and configuration files for your application. You can also use *CompuCAM*, Compumotor's Computer Aided Motion program, to generate complex motion profiling programs. By downloading program or path files and controlling their execution, the Motion Toolbox application acts as a supervisor over the 6000 controller. This approach offers excellent performance for demanding applications.

 **Complete file path** is the path name of the file. If file path is empty (default value) or is Not A Path, the VI displays a File dialog box from which you can select a file.

6000 command reference: N/A

## Enable Communications Tracing



Enables tracing of 6000 communications. When enabled, tracing will dump all command and query information to the file specified by *trace path* (see *Configure Communications Tracing VI*). This is a very effective tool for debugging Motion Toolbox applications.

Each line of the trace file dump is pre-pended by the address of the command or query destination and a operation specifier. The operation specifiers are:

- S: Send.
- R: Receive.
- Q: Question -- the command part of a query operation.
- A: Answer -- the response part of a query operation.

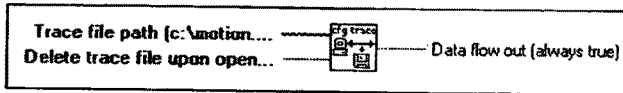
**TF** Enable Tracing?

True: Enable 6000 communications tracing.  
False: Disable 6000 communications tracing.

**TF** Data flow out is always true. It can be used to force correct sequencing of this VI.

6000 command reference: N/A

### Configure Communications Tracing



Sets the trace path used for communications tracing. This VI also specifies whether to delete the trace file indicated by *Trace file path* upon executing *Open AT6400* or *Open 6200*.

**Note:** Very good performance is achieved by directing the trace file to a RAM disk.

**Trace file path** specifies the path of the file to use for dumping communications information during tracing.

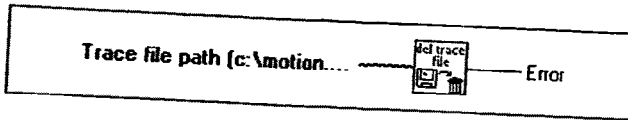
**TF** Delete trace file upon open?

True: Delete trace file upon executing *Open 6200* or *Open AT6400*.  
False: Do not delete trace file.


**TF** Data flow out is always true. It can be used to force correct sequencing of this VI.

6000 command reference: N/A

## Delete Trace File

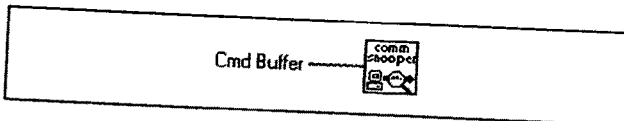


Deletes the specified communications trace file.


 Trace file path specifies the trace file to delete.

6000 command reference: N/A

## Command Snooper



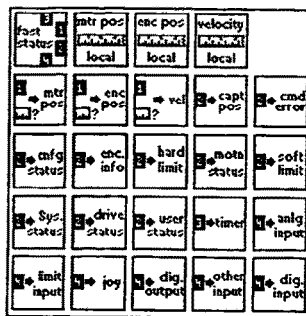
Allows viewing of commands sent to the 6000 controller by Motion Toolbox. It does not display fast status, transfer, or query information. To use Command Snooper to debug programs, open the VI and move it to a convenient location. The VI will display the commands as they are sent to the 6000 controller. It may be helpful to put LabVIEW in debug mode to slow down the execution of your program. This VI is for debugging only and is not intended for use in programs.

 Cmd Buffer is only used by Motion Toolbox. Do not use this VI in a program.

6000 command reference: All 6000 commands used by Motion Toolbox can be displayed.

## Fast Status VIs

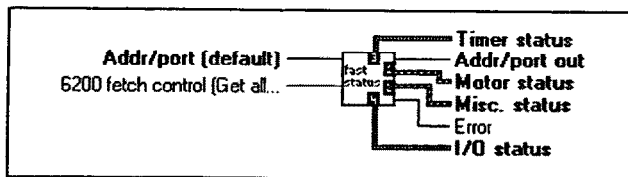
This chapter contains descriptions of the fast status VIs included with Motion Toolbox. The figure below displays the *Fast Status* function palette.



*Fast Status Function Palette*

## Fast Status VI Descriptions

### Get Fast Status



Reads fast status information from the specified 6000 controller. The "Fetch control" input specifies what clusters to actually retrieve and is only relevant when using 62XX products.

To use this VI effectively, place it in a while loop that iterates at a rate that provides an appropriate status update for the application. Feed the outputs of

Get Fast Status to the appropriate parsing VIs. Place a *Get Fast Status* VI in each while loop of an application that needs status information. Alternatively, use global variables to "broadcast" the fast status information to other loops. Avoid using multiple instances of this VI within a single while loop as this will incur unnecessary overhead.

**132** **Fetch Control** specifies what status clusters to read from the 62XX indexer. At 9600 baud, it typically takes between 20 and 35 milliseconds to fetch a fast status block (cluster) from a 6200 indexer. By fetching only the status clusters that are needed, application performance is enhanced.

- Bit 1: Enable/disable fetching of Motor status. This cluster contains all motor and encoder positions and velocity information.
- Bit 2: Enable/disable fetching of Misc. status. This cluster contains the controller's axis status, system status and user status information.
- Bit 3: Enable/disable fetching of Timer status. This cluster contains values for the programmable timer and the two millisecond time frame mark.
- Bit 4: Enable/disable fetching of I/O status. This cluster retrieves the current physical values for the inputs and outputs.

For example, to configure *Get Fast Status* to fetch only motor and I/O status, set *Fetch control* to 1001 binary or 9 decimal.

Note: This input is ignored when *Get Fast Status* is used with an AT6400.

**133** **Motor status** is a cluster of motor related parameters.

**134** **Misc. status** is a cluster of miscellaneous parameters.

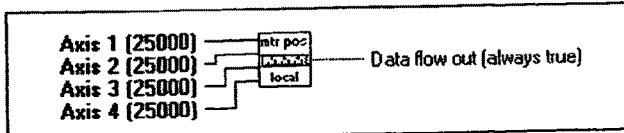
**135** **Timer status** is a cluster of timer related parameters.

**136** **I/O status** is a cluster of I/O related parameters.

6000 command reference: TAS, TSS, TUS



### Set Motor Pos Local Scaling



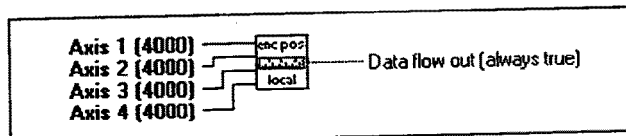
Sets the local scaling factors used by the *Motor Position Parse VI*. Use this VI to scale distances to units convenient for your application. For example, if your step motor is set to a resolution of 25,000 steps per revolution and the motor is attached to a linear table with a five turn-per-inch leadscrew, the motor will travel 125,000 steps per inch (25,000 steps/rev \* 5 rev/in). To display inches, set the local scaling to 125,000 for the appropriate axis and the information from the *Motor Position Parse VI* will be formatted to inches.

**IS2** Axis 1-4 specify the motor position scaling factors for the respective axis. Each axis can have a unique scale factor.

**TF** Data flow out is always true. It can be used to force correct sequencing of this VI.

6000 command reference: N/A

### Set Encoder Pos Local Scaling



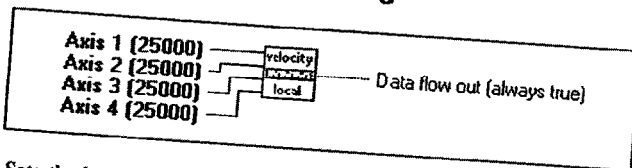
Sets the local scaling factors used by the *Encoder Position Parse VI*. The scale factors are in post-quadrature counts. Use this VI to scale encoder distances to units convenient for your application. For example, if your encoder is mounted on a linear table with a resolution of 10,000 counts per inch, the scale factor for that axis should be set to 10,000 to scale the encoder information to inches.

**IS2** Axis 1-4 specify the encoder position scaling factors for the respective axis. Each axis can have a unique scale factor.

**TF** Data flow out is always true. It can be used to force correct sequencing of this VI.

6000 command reference: N/A

### Set Velocity Local Scaling



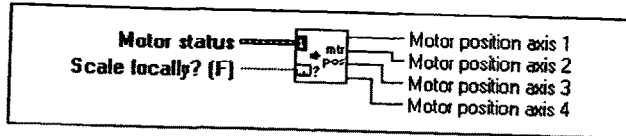
Sets the local scaling factors used by the *Velocity Parse* VI. Use this VI to scale velocities to units convenient for your application. For example, if your step motor is set to a resolution of 25,000 steps per revolution and the motor is attached to a linear table with a five turn-per-inch leadscrew, the motor will travel 125,000 steps per inch (25,000 steps/rev \* 5 rev/in). To display velocity in inches per second, set local scaling to 125,000 for the appropriate axis and the respective output of the *Velocity Parse* VI will be formatted to inches per second.

**IS2** Axis 1-4 specify the velocity scaling factors for the respective axis. The default input scales velocity information to revolutions per second when used with a 25,000 step-per-revolution step motor system. Each axis can have a unique scale factor.

**TF** Data flow out is always true. It can be used to force correct sequencing of this VI.

6000 command reference: N/A

## Motor Position Parse



Parses the *Motor status* output of the *Get Fast Status VI* into motor positions for each axis. If *Scale locally* is enabled, the motor positions are divided by the respective scale factor set by the *Set Motor Pos. Local Scaling VI*.

**ENB** Motor status is the *Motor status* output of the *Get Fast Status VI*.

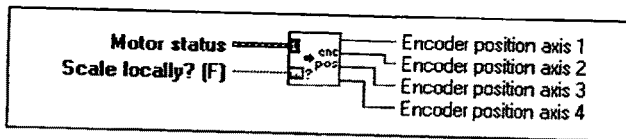
**YF** Scale locally?

True: Divide motor positions by respective local scaling factor.  
False: Do not scale.

**DBL** Motor position axis 1-4 are the motor positions for each axis in motor steps, or, if local scaling is enabled, in scaled motor steps.

6000 command reference: N/A

## Encoder Position Parse



Parses the *Motor status* output of the *Get Fast Status VI* into encoder positions for each axis. If *Scale locally* is enabled, the encoder positions are divided by the respective scale factor set by the *Set Encoder Pos. Local Scaling VI*.

**ENB** Motor status is the *Motor status* output of the *Get Fast Status VI*.

**TF** Scale locally?

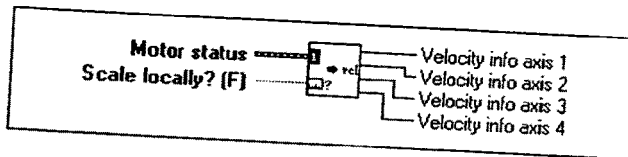
True: Divide encoder positions by respective local scaling factor.

False: Do not scale.

**DBL** Encoder position axis 1-4 are the encoder positions for each axis in encoder counts, or, if local scaling is enabled, in scaled encoder counts.

6000 command reference: N/A

### Velocity Parse



Parses the *Motor status* output of the *Get Fast Status VI* into velocities for each axis. If *Scale locally* is enabled, the velocities are divided by the respective scale factor set by the *Set Velocity Local Scaling VI*.

**DBL** Motor status is the *Motor status* output of the *Get Fast Status VI*.

**TF** Scale locally?

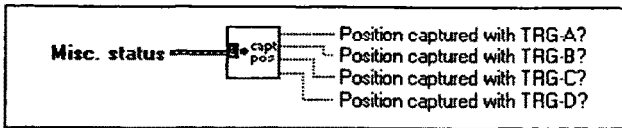
True: Divide velocities by respective local scaling factor.

False: Do not scale.

**DBL** Velocity info axis 1-4 are the respective velocities for each axis in steps/second, or, if local scaling is enabled, in scaled steps/second.

6000 command reference: N/A

### Pos. Captured Status Parse



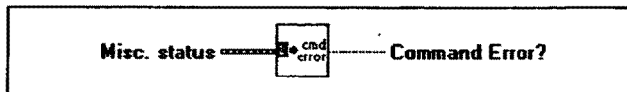
Parses the *Misc. status* output of the *Get Fast Status VI* and returns booleans to indicate if a trigger input has captured its respective position. To retrieve the captured position, use the *Transfer Captured Positions VI*.

**POS** *Misc. status* is the *Misc. status* output of the *Get Fast Status VI*.

**TF** Position captured with TRG-A-D? True if a position has been captured for the respective trigger input.

6000 command reference: TSS

### Command Error Parse



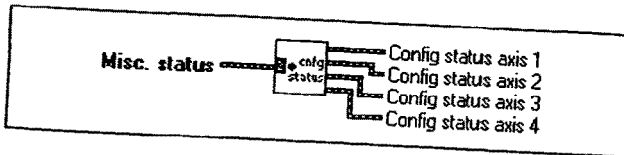
Parses the *Misc. status* output of the *Get Fast Status VI* and returns the command error status of the 6000 controller. This command error bit remains set until cleared by a TCMDER query (see *Miscellaneous Transfer VI*).

**POS** *Misc. status* is the *Misc. status* output of the *Get Fast Status VI*.

**TF** Command Error? True if the controller has received an unrecognized command since the last TCMDER query, power on, or reset. The *Miscellaneous Transfer VI* (issues a TCMDER query) will return the command string unrecognized by the controller as well as reset the command error flag.

6000 command reference: TCMDER

## Configuration Status Parse



Parses the *Misc. status* output of the *Get Fast Status VI* and returns the configuration status of each axis.

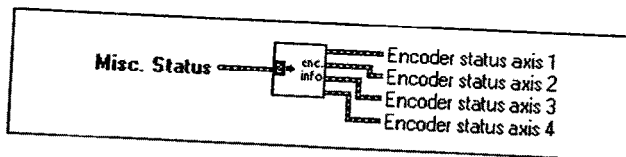
**FIG 208** *Misc. status* is the *Misc. status* output of the *Get Fast Status VI*.

**FIG 209** *Config status axis 1-4* is a cluster of six booleans that report the axis specific status of the following parameters:

- In absolute mode (true) / In incremental mode
- In continuous (true) / in preset mode
- In jog mode (true) / Not in jog mode
- In joystick mode (true) / Not in joystick mode
- In encoder step mode (true) / In motor step mode
- Position Maintenance ON (true) / OFF

6000 command reference: TAS

## Encoder Feedback Status Parse



Parses the *Misc. status* output of the *Get Fast Status VI* and returns the encoder feedback status for each axis.

**FIG 210** *Misc. status* is the *Misc. status* output of the *Get Fast Status VI*.

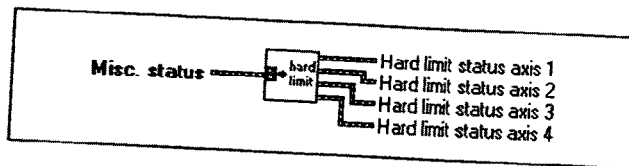
**ENC** Encoder status axis 1-4 is a cluster of two booleans that report the axis specific status of the following parameters:

*Stall Detected?* Is true if stall detected.

*Within Deadband?* Is true if motor is within deadband.

6000 command reference: TAS

### Hard Limit Status Parse



Parses the *Misc. status* output of the *Get Fast Status VI* and returns the hard limit status for each axis.

**ENC** *Misc. status* is the *Misc. status* output of the *Get Fast Status VI*.

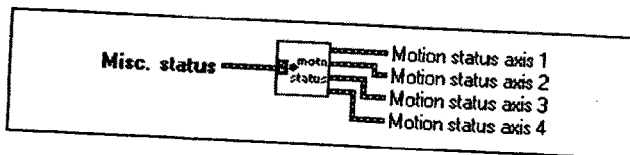
**ENC** Hard limit status axis 1-4 is a cluster of two booleans that report the axis specific status of the following parameters:

*CW* is true if CW limit has been encountered.


*CCW* is true if CCW limit has been encountered.


6000 command reference: TAS

### Motion Status Parse



Parses the *Misc. status* output of the *Get Fast Status VI* and returns the motion status for each axis.

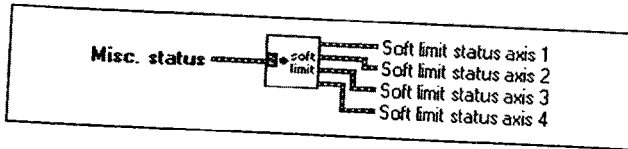
 **Misc. status** is the *Misc. status* output of the *Get Fast Status VI*.

 **Motion status axis 1-4** is a cluster of five booleans that report the axis specific status of the following parameters:


- Moving?* True if axis is moving.
- Direction = CCW?* True if axis direction is CCW.
- Accelerating?* True if axis is accelerating.
- At velocity?* True if axis is at velocity.
- Home successful?* True if axis home was successful.


6000 command reference: TAS

### Soft Limit Status Parse



Parses the *Misc. status* output of the *Get Fast Status VI* and returns the soft limit status for each axis.

 **Misc. status** is the *Misc. status* output of the *Get Fast Status VI*.

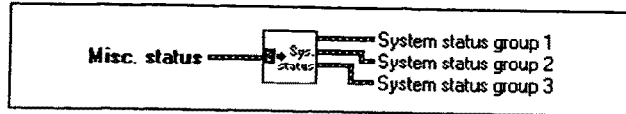
 **Soft limit status axis 1-4** is a cluster of two booleans that report the axis specific status of the following parameters:

- *CW*. True if soft, clockwise limit was encountered.
- *CCW*. True if soft, counter-clockwise limit was encountered.

6000 command reference: TAS



## System Status Parse



Parses the *Misc. status* output of the *Get Fast Status VI* and returns the system status.

**508** *Misc. status* is the *Misc. status* output of the *Get Fast Status VI*.

**509** **System status group 1** is a cluster of eight booleans that report the system status of the following parameters (true = yes):

- *System ready?*
- *Buffer full? (AT6400 only)*
- *Executing a program?*
- *Immediate command?* Set true if last command was immediate)
- *In ASCII mode?*
- *In echo mode?*
- *Defining a program?*
- *In trace mode?*

**510** **System status group 2** is a cluster of eight booleans that report the system status of the following parameters (true = yes):

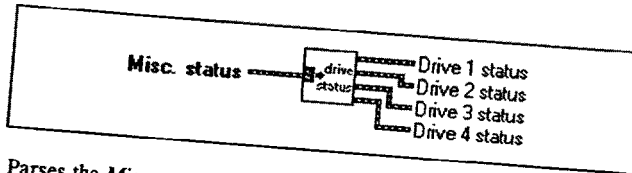
- *In step mode?*
- *In translation mode (6400)?*
- *Command error occurred?* This boolean (status bit) is cleared when the TCMER command (via the *Miscellaneous Transfer VI*) is issued.
- *Break point active?*
- *Pause active?*
- *Wait active?*
- *Monitoring on condition?*
- *Waiting for data?*

**6002** System status group 3 is a cluster of eight booleans that report the system status of the following parameters (true = yes):

- Loading thumbwheel data?
- External program select mode?
- Dwell in progress (T command)?
- Waiting for RP240 data? (62XX)?
- RP240 connected? (62XX)?
- Non-volatile memory error (62XX)?
- Servo data transmission? (6250)
- Reserved

6000 command reference: TSS

### Drive Status Parse



Parses the *Misc. status* output of the *Get Fast Status VI* and returns the drive status for each axis.

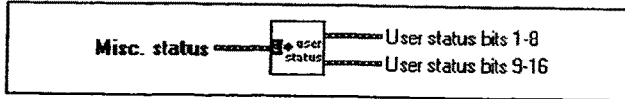
**6003** *Misc. status* is the *Misc. status* output of the *Get Fast Status VI*.

**6004** *Drive 1-4 status* is a cluster of two booleans that report the axis specific status of the following parameters:

- *Shutdown?* True if drive is shut down.
- *Fault?* True if drive fault has occurred.

6000 command reference: TAS

## User Status Parse

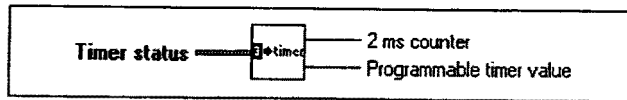


Parses the *Misc. status* output of the *Get Fast Status VI* and returns user status information.

- 208** *Misc. status* is the *Misc. status* output of the *Get Fast Status VI*.
- 208** *User status bits 1-8* is a cluster of eight booleans that report the status of the first byte of the user status.
- 208** *User status bits 9-16* is a cluster of eight booleans that report the status of the second byte of the user status.

6000 command reference: TUS, INDUSE, INDUST

## Timer Status Parse

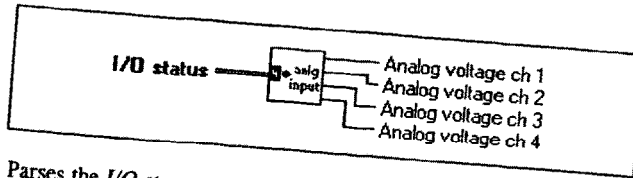


Parses the *Timer status* output of the *Get Fast Status VI* and returns the 2 ms time frame counter and the programmable timer values. The programmable timer is started and stopped with the *Start 6000 Timer* and *Stop 6000 Timer* VIs.

- 208** *Timer status* is the *Timer status* output of the *Get Fast Status VI*.
- U16** *2 ms counter* contains the value of the counter. Range is 0 to 65,535 counts. The counter rolls over after 65,535.
- U32** *Programmable timer value* contains the value of the programmable counter. Range is zero to 999,999,999 milliseconds.

6000 command reference: TTIM

## Analog Input Parse



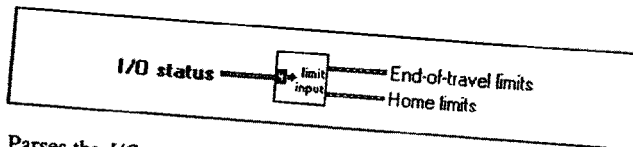
Parses the *I/O status* output of the *Get Fast Status VI* and returns the values of the analog inputs on the joystick connector.

**206** I/O status is the *I/O status* output of the *Get Fast Status VI*.

**561** Analog voltage ch 1-4 contains the value of the respective analog input. The range is 0 to 255. This is the raw value from the eight bit, analog-to-digital converter where 0 corresponds to 0 volts and 255 corresponds to 2.5 volts.

6000 command reference: TANV

## Limit Input Status Parse



Parses the *I/O status* output of the *Get Fast Status VI* and returns the status of all end-of-travel and homing limits.

**505** I/O status is the *I/O status* output of the *Get Fast Status VI*.

**208** End-of-travel limits is a cluster of eight booleans that report the *actual physical status* of the end-of-travel limits for all axes. The booleans order is CW limit then CCW limit for axis one and is repeated for axes two through four.

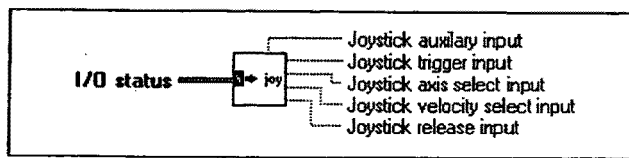
- *CW axis n*. True if clockwise limit is currently active.
- *CCW axis n*. True if counter-clockwise limit is currently active.

**True** Home limits is a cluster of four booleans that report the *actual physical status* of the home limits for all axes. The boolean order is from axis one to four.

- *Axis n.* True if home limit is currently active.

6000 command reference: TLIM

### Joystick Status Parse



Parses the *I/O status* output of the *Get Fast Status VI* and returns the status of the actual physical state of various joystick inputs.

**True** *I/O status* is the *I/O status* output of the *Get Fast Status VI*.

**TF** Joystick auxiliary input is true if pin 19 of the joystick connector is currently active.

**TF** Joystick trigger input is true if pin 18 of the joystick connector is currently active.

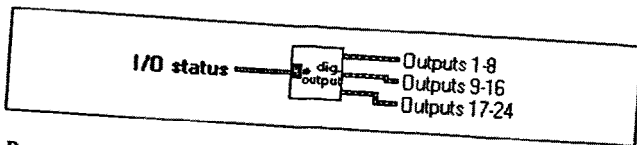
**TF** Joystick axis select input is true if pin 15 of the joystick connector is currently active.

**TF** Joystick velocity select input is true if pin 16 of the joystick connector is currently active.

**TF** Joystick release input is true if pin 17 of the joystick connector is currently active.

6000 command reference: TINO

## Digital Output Status Parse



Parses the *I/O status* output of the *Get Fast Status VI* and returns the status of all programmable digital outputs.



*I/O status* is the *I/O status* output of the *Get Fast Status VI*.



*Outputs 1-8* is a cluster of eight booleans that indicate the status of the first byte of programmable outputs. A given boolean is true if the output is currently active.



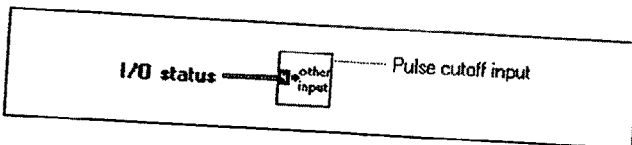
*Outputs 9-16* is a cluster of eight booleans that indicate the status of the second byte of programmable outputs. A given boolean is true if the output is currently active.



*Outputs 17-24* is a cluster of eight booleans that indicate the status of the third byte of programmable outputs. A given boolean is true if the output is currently active.

6000 command reference: TOUT

## Other Input Parse



Parses the *I/O status* output of the *Get Fast Status VI* and returns the status of the pulse cutoff input. This input must be grounded (active) before any motion can take place.



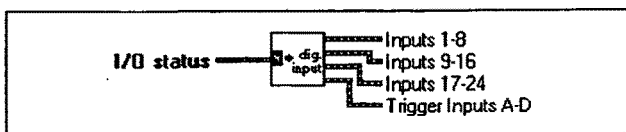
*I/O status* is the *I/O status* output of the *Get Fast Status VI*.

**TF** Pulse cutoff is true if the pulse cutoff input is grounded.

**Note:** The pulse cutoff input must be grounded (active) before any motion can take place.

6000 command reference: TINO

### Digital Input Parse



Parses the *I/O status* output of the *Get Fast Status VI* and returns the status of all digital and trigger inputs.

**IO** *I/O status* is the *I/O status* output of the *Get Fast Status VI*.

**IO** *Inputs 1-8* is a cluster of eight booleans that indicate the status of the first byte of programmable inputs. A given boolean is true if the input is currently active.

**IO** *Inputs 9-16* is a cluster of eight booleans that indicate the status of the second byte of programmable inputs. A given boolean is true if the input is currently active.

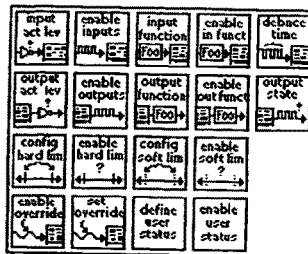
**IO** *Inputs 17-24* is a cluster of eight booleans that indicate the status of the third byte of programmable inputs. A given boolean is true if the input is currently active.

**IO** *Trigger Inputs A-D* is a cluster of four booleans that report the status of the trigger inputs. A given boolean is true if the trigger input is currently active.

6000 command reference: TIN

## I/O & Limit VIs

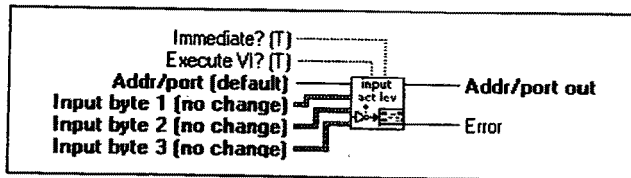
This chapter contains descriptions of the I/O and limit VIs included with Motion Toolbox. The figure below displays the *I/O & Limits* function palette.



*I/O & Limits Function Palette*

## I/O & Limit VI Descriptions

### Set 6000 Input Active Level



Defines the active state of all programmable inputs. The active level can be defined for each input individually. The 6000 series default state is active low. Home and limit active levels are set using the *Configure Home* and *Set Hard Limit Active Level* VIs respectively.

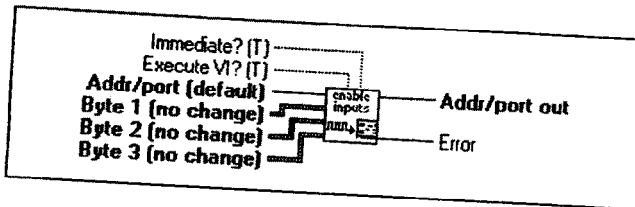


**ENEN** Input byte 1-3 are clusters of eight, 32-bit integers that specify the active state of the corresponding input. Each cluster is ordered from the least significant input to the most significant.

- 0: Set input to active low.
- 1: Set input to active high.
- 2: Do not change current setting.

6000 command reference: INLVL

### Enable 6000 Inputs



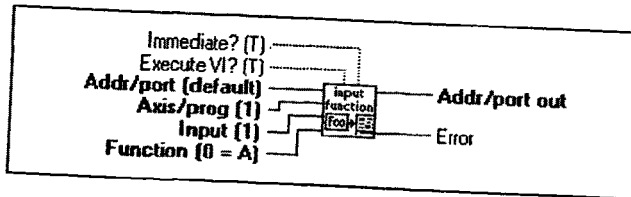
Enables or disables the specified inputs. This VI allows inputs to be disabled and forced to a high or low state. It may be used for debugging purposes. It is generally not used within a program. The default state for each input is enabled. When the input is enabled, the function programmed for that input will be active.

**ENEN** Input byte 1-3 are clusters of eight, 32-bit integers that specify the enable/disable state of the corresponding input. Each cluster is ordered from the least significant input to the most significant.

- 0: Disable input and leave off.
- 1: Disable input and leave on.
- 2: Do not change enable setting.
- 3: Enable the input.

6000 command reference: INEN

## Set 6000 Input Function



Defines the function corresponding to the specified input and axis number (if required). Each input may be assigned a unique function. The default condition is function 0 for all inputs. To enable the functions defined in this VI, use the *Enable 6000 Input Functions VI*.

**I32** **Axis/prog** specifies the axis number if required by the function or in the case of function 15, it specifies the program number to execute.

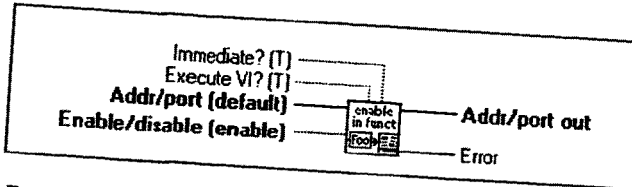
**I32** **Input** specifies the input number to define.

**I32** **Function** specifies the function to assign to the input and axis.

- 0: No special function
- 1: BCD program select
- 2: Kill
- 3: Stop
- 4: Pause/Continue
- 5: User fault
- 6: Reserved
- 7: Trigger interrupt (Position Capture and Registration)
- 8: Interrupt to PC-AT
- 9: Jog CW
- 10: Jog CCW
- 11: Jog speed select
- 12: Reserved
- 13: Reserved
- 14: Reserved
- 15: Program select

6000 command reference: INFNC

## Enable 6000 Input Functions



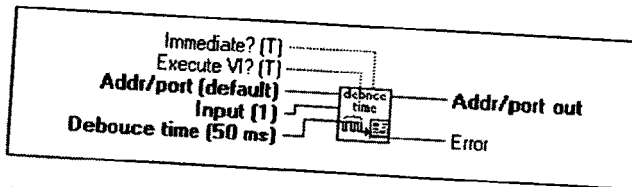
Enables or disables the drive fault input and input functions defined by the *Set 6000 Input Function* VI. All input functions defined with the *Set 6000 Input Function* VI will have no effect unless enabled by this VI.

**TF** Enable/disable.

True: Enable input functions and drive fault input.  
False: Disable input functions and drive fault input.

6000 command reference: INFEN, INFNC, DRFLVL

## Set 6000 Input Debounce Time



Sets the input debounce time for all of the general purpose inputs or assigns a unique debounced time to each of the trigger inputs. The range is 2 to 250 milliseconds in even increments. The 6000 series defaults are 4 ms for the general purpose inputs and 50 ms for the trigger inputs.

**IS2** Input.

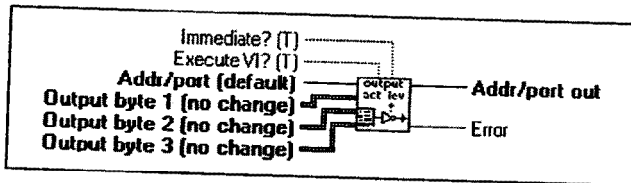
1-24: Sets the debounce time for all general purpose inputs regardless of number chosen (1-24).  
25-28: Sets the debounce time for the corresponding trigger input.

**I32** **Debounce time.** For a general purpose input, *Debounce time* specifies the amount of time the input must be held in a certain state before the 6000 Series controller recognizes it. This directly affects the rate at which the inputs can change state and be recognized.

For a trigger input, *Debounce time* specifies the time required between a trigger's initial active transition and its secondary active transition. This allows rapid recognition of a trigger, but prevents subsequent bouncing of the input causing a false position capture or registration move.

6000 command reference: INDEB

### Set 6000 Output Active Level



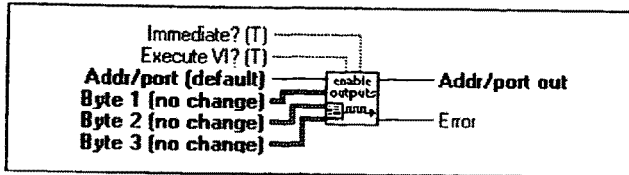
Defines the active state of all programmable outputs. The default state is active low. The active level can be defined for each output individually.

**502** **Output byte 1-3** are clusters of eight, 32-bit integers that specify the active state of the corresponding output. Each cluster is ordered from the least significant input to the most significant. The eight clustered integers are interpreted as follows.

- 0: Set output to active low.
- 1: Set output to active high.
- 2: Do not change current setting.

6000 command reference: OUTLVL

## Enable 6000 Outputs



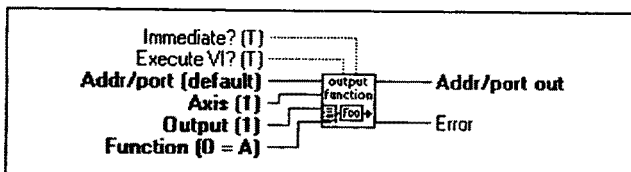
Enables or disables the specified outputs. It is used for troubleshooting and initial start-up testing. It is generally **not** used within a program. The 6000 series default state for each output is enabled.

**ES1** Byte 1-3 are clusters of eight, 32-bit integers controlling the enable/disable state of the respective output. Each cluster is ordered from the least significant output to the most significant. The eight clustered integers are interpreted as follows.

- 0: Disable output and set to off.
- 1: Disable output and set to on.
- 2: Do not change current setting.
- 3: Enable output.

6000 command reference: OUTEN

## Set 6000 Output Function



Defines the function corresponding to the specified output and axis number (if required). Each output may be assigned a unique function. The default condition is function 0 for all of the outputs. To enable the functions by this VI, use the *Enable 6000 Output Functions VI*.

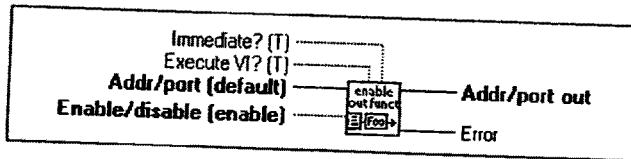
**ES2** Output specifies the output number to define.

**132** Function specifies the function to assign to the output and axis.

- 0: Programmable output
- 1: Moving/Not Moving Axis, axis specifier optional
- 2: Program in progress
- 3: At limits, hard or soft, axis specifier optional
- 4: Stall indicator, axis specifier optional -- stepper products only
- 5: Fault output (indicates drive or user fault)
- 6: Position error exceeds maximum limit set with SMPER command -- servo products only.
- 7: Output on position -- servo products only

6000 command reference: OUTFNC

### Enable 6000 Output Functions



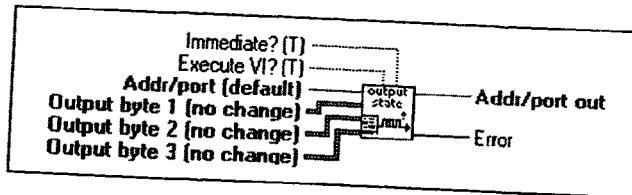
Enable or disables the output functions defined by the *Set 6000 Output Function VI*. If output functions are disabled, the outputs can be used as programmable outputs only.

**TF** Enable/disable.

- True: Enable output functions.
- False: Disable output functions.

6000 command reference: OUTFEN

## Set 6000 Output States



Sets the state of outputs configured as programmable. In order for this VI to affect the state of an output, that output must be defined as a programmable using the *Set 6000 Output Function VI*.

**IS2** **Output byte 1-3** are clusters of eight, 32-bit integers that specify the state of the corresponding output defined as a programmable output. Each cluster is ordered from the least significant input to the most significant. The eight clustered integers are interpreted as follows.

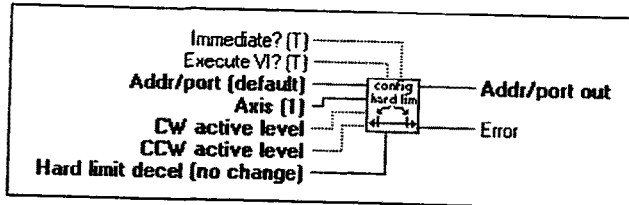
- 0: Set output off.
- 1: Set output on.
- 2: Do not change setting.

**Note:** The integers of the clusters **Output byte 1-3** do not necessarily correspond one-to-one with the physical 6000 outputs; the integers correspond to outputs configured as programmable outputs.

**For example:** If only outputs 3 and 5 were defined as programmable outputs, then the first two integers (not the 3rd and 5th) of the **Output byte 1** cluster would affect the state of the outputs.

6000 command reference: OUT

## Configure Hard Limits



Defines the active state of the limit inputs and deceleration used when a hard limit is encountered. For safety reasons, the 6000 series default state is active low.

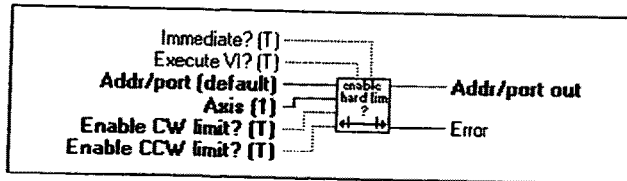
**TF** CW/CCW active level.

True: Limit is active high.  
False: Limit is active low.

**DBL** **Hard Limit Decel** defines the deceleration rate in the event an end-of-travel limit is encountered. The range is 0.00025 - 24,999,999 units/sec<sup>2</sup>. The 6000 series default value is 100 units/sec<sup>2</sup>.

6000 command reference: LHLVL, LHAD

## Enable Hard Limits



Enables or disables the use of the hard wired limits. With the limits disabled, motion will not be restricted. When a specific limit is enabled, and the limits wiring for the enabled limit is physically active according to settings issued by the *Set Hard Limit Active Level VI*, motion will be restricted. The 6000 series default condition is enabled.

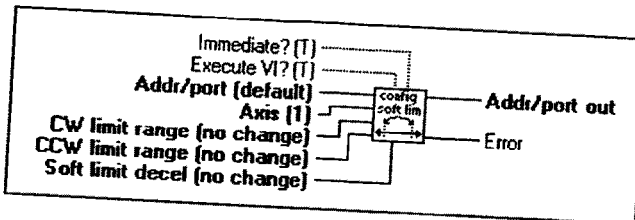


**[TF]** Enable CW/CCW limit

True: Enable limit.  
False: Disable limit.

6000 command reference: LH

### Configure Soft Limits



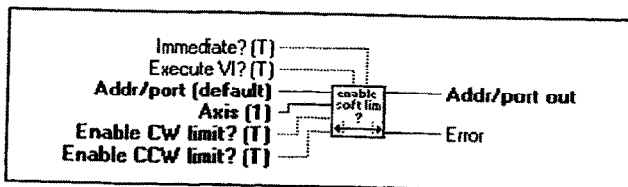
Defines the range in absolute distance units within which the specified axis can move unrestricted. Should the axis move beyond the specified range, the axis will decelerate to a stop at the rate defined by the *Soft limit decel* input. **Always** set the CW limit range greater than the CCW limit range.

**[DBL]** CW/CCW limit range defines the maximum absolute position the axis can travel unrestricted. The range is -999,999,999 to +999,999,999 units.

**[DBL]** Soft Limit Decel defines the deceleration rate in the event an end-of-travel limit is encountered. The range is 0.00025 to 24,999,999 units/sec<sup>2</sup>. The 6000 series default value is 100 units/sec<sup>2</sup>.

6000 command reference: LSCW, LSCCW, LSAD

### Enable Soft Limits



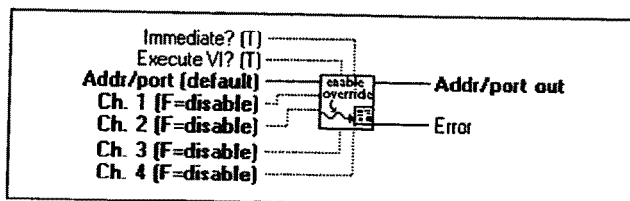
Enables or disables the soft limits on the specified axis. When disabled, motion will not be restricted by the soft limits. When a specific limit is enabled, and the absolute limit defined in the *Set Soft Limit Range VI* is exceeded, motion will stop at the pre-defined deceleration rate.

**TF** Enable CW/CCW limit.

True: Enable soft limit.  
False: Disable soft limit.

6000 command reference: LS

### Enable Analog Input Override



Enables or disables software overriding of the analog inputs. If enabled, the analog input values are as specified by the *Set Analog Input Override Voltage VI*. If disabled, the analog input value are as determined by the actual hardware input voltages on the joystick connector.

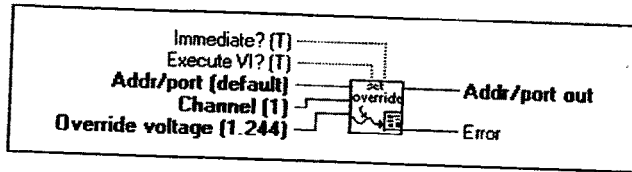
The joystick release input (pin #17 on the Joystick connector) is not monitored when override is enabled for any analog input channel. Thus, you can enter the joystick mode and simulate joystick operations without having to wire that input.

**TF** Ch 1-4. (Product dependent)

True: Enable analog input override for respective channel.  
False: Disable analog input override for respective channel.

6000 command reference: ANVOEN

### Set Analog Input Override Voltage



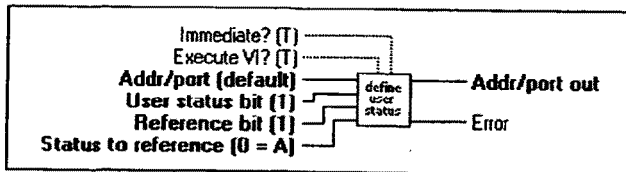
After enabling analog input override with the *Enable Analog Input Override* VI, this VI will set the values of the analog inputs. The values set by this VI will be used for all commands and functions that reference the analog inputs, but only those channels for which analog input override has been enabled.

**132** **Channel** specifies the analog input channel to set. Valid channels are one to four and are product dependent.

**DBL** **Override voltage** specifies the value for the analog input channel. The range is 0-2.5 volts. The 6000 series default value is 1.244 volts.

6000 command reference: ANVO

## Define 6000 User Status



Defines the bits of the user status word. The user status word is a configurable combination of status bits referenced from other 6000 status words. This facility allows you to create your own application-specific status word. This status word is retrieved with the *User Status Parse VI*.

**152** **User status bit** specifies the user status bit to define. Valid range is 1 to 16.

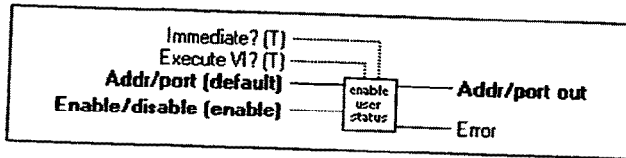
**152** **Reference bit.** Specifies the system status, axis status, input #, or interrupt status bit to reference. Valid range is 1 to 32.

**152** **Status to reference.** Specifies the status information to reference.

- 0: A - axis status for axis 1
- 1: B - axis status for axis 2
- 2: C - axis status for axis 3 (AT6400 only)
- 3: D - axis status for axis 4 (AT6400 only)
- 4: E - reserved
- 5: F - reserved
- 6: G - reserved
- 7: H - reserved
- 8: I - system status
- 9: J - input status
- 10: K - interrupt status (AT6400 only)

6000 command reference: INDUST

## Enable 6000 User Status



Enables or disables the update of the user status word defined by the *Define 6000 User Status VI*.

**TF** Enable/disable.

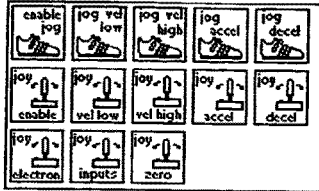
True: Enable update of user status.

False: Disable update of user status.

6000 command reference: INDUSE, INDUST

# Jogging & Joystick VIs

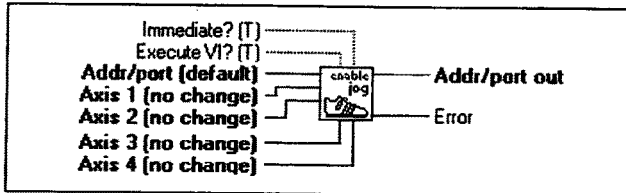
This chapter contains descriptions of the jogging and joystick VIs included with Motion Toolbox. The figure below displays the *Jogging & Joystick* function palette.



*Jogging & Joystick Function Palette*

## Jogging and Joystick VI Descriptions

### Enable Jog Mode



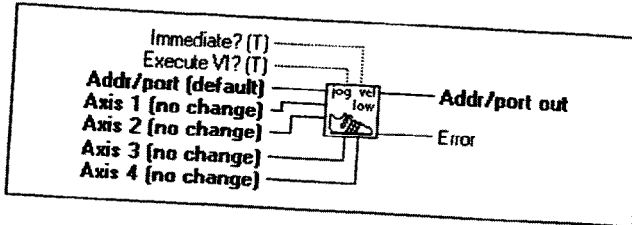
Enables jog mode on the specified axes. Once jog mode is enabled, the jog inputs can be used to produce motion on the specified axis. Inputs are configured as jog inputs using the *Set 6000 Input Function VI*.

**TS2** Axis 1-4 enables/disables the jog mode for the respective axis.

- 0: Disable jog mode.
- 1: Enable jog mode.
- 2: Do not change current setting.

6000 command reference: JOG

## Set Jog Velocity Low

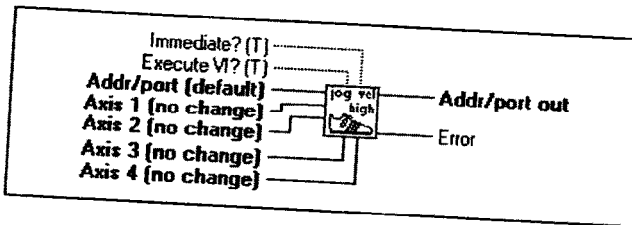


Specifies the velocity to be used upon receiving a jog input with the jog speed select input low (OFF). Inputs can be configured as a jog speed input using the *Set 6000 Input Function VI*.

**DBL** Axis 1-4 defines the low jog velocity for the respective axis. Valid range is 0 to 1,600,000 steps/sec.

6000 command reference: JOGVL

## Set Jog Velocity High

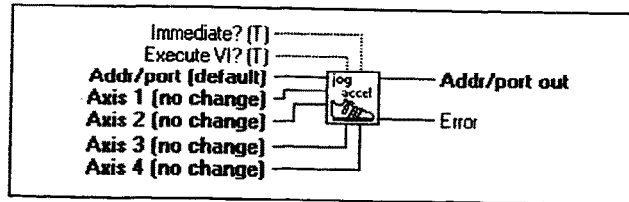


Specifies the velocity to be used upon receiving a jog input with the jog speed select input high (ON). Inputs can be configured as a jog speed input using the *Set 6000 Input Function VI*.

**DBL** Axis 1-4 defines the high jog velocity for the respective axis. Valid range is 0 to 1,600,000 steps/sec.

6000 command reference: JOGVH

### Set Jog Acceleration

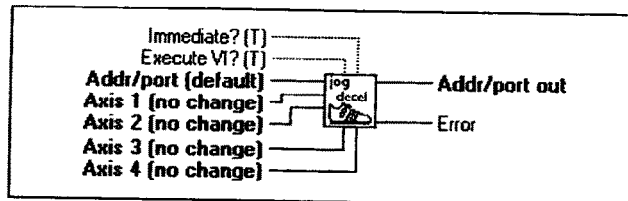


Specifies the acceleration used during jog mode.

**DBL** Axis 1-4 defines the jog acceleration for the respective axis. Valid range is 0.00025 to 24,999,999 steps/sec<sup>2</sup>.

6000 command reference: JOGA

### Set Jog Deceleration



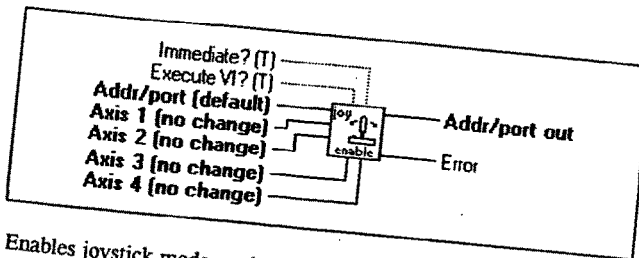
Specifies the deceleration used during jog mode.

**DBL** Axis 1-4 defines the jog deceleration for the respective axis. Valid range is 0.00025 to 24,999,999 steps/sec<sup>2</sup>.

6000 command reference: JOGAD



## Enable Joystick Mode



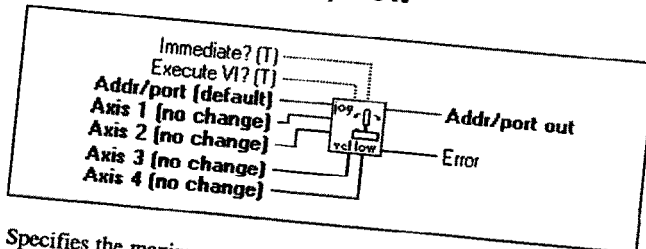
Enables joystick mode on the specified axes. Once joystick mode is enabled, the analog inputs can be used to produce motion on a specific axis. Motion will be directly proportional to the voltage on the analog inputs. Command processing in the controller stops until the joystick release becomes active.

**ES2** Axis 1-4 enables/disables joystick mode for the respective axis.

- 0: Disable joystick mode.
- 1: Enable joystick mode.
- 2: Do not change current setting.

6000 command reference: JOY

## Set Joystick Velocity Low

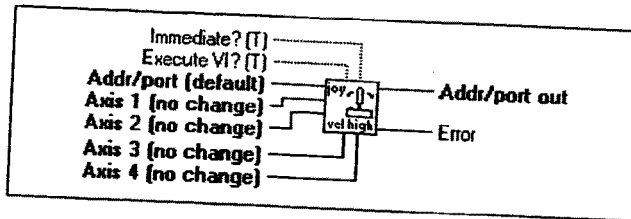


Specifies the maximum velocity that can be obtained at full deflection (0 VDC or 2.5 VDC) during joystick mode while the joystick velocity select input on the joystick connector is low.

**DB1** Axis 1-4 defines the low joystick velocity for the respective axis. Valid range is 0.00020 to 1,600,000 steps/sec.

6000 command reference: JOYVL

### Set Joystick Velocity High

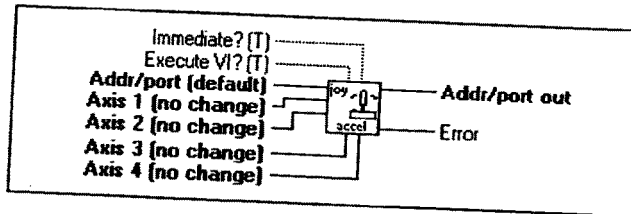


Specifies the maximum velocity that can be obtained at full deflection (0 VDC or 2.5 VDC) during joystick mode while the joystick velocity select input on the joystick connector is high.

**DBL** Axis 1-4 defines the high joystick velocity for the respective axis. Valid range is 0.00020 to 1,600,000 steps/sec.

6000 command reference: JOYVH

### Set Joystick Acceleration

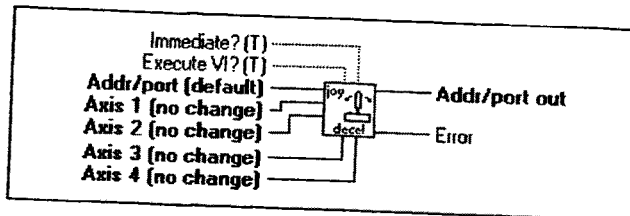


Specifies the acceleration used during joystick mode.

**DBL** Axis 1-4 defines the joystick acceleration for the respective axis. Valid range is 0.07500 to 24,999,999 steps/sec<sup>2</sup>.

6000 command reference: JOYA

## Set Joystick Deceleration

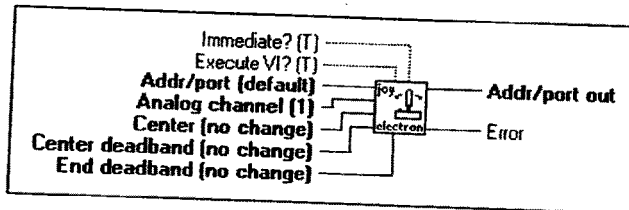


Specifies the deceleration used during joystick mode.

**DBL** Axis 1-4 defines the joystick deceleration for the respective axis. Valid range is 0.07500 to 24,999,999 steps/sec<sup>2</sup>.

6000 command reference: JOYAD

## Setup Joystick Electronics



Defines the electronic characteristics of a given analog channel for use in joystick mode. The 6000 series joystick inputs are eight-bit ADCs with a 0 to 2.5 volt input.

**TS2** Analog channel specifies the analog input channel to define. Valid range is 0 to 4, product dependent.

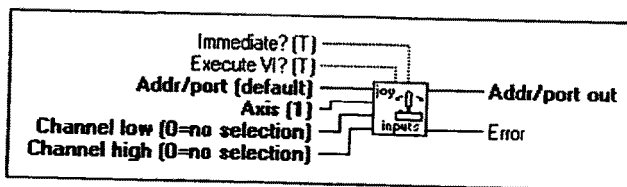
**DBL** Center defines the voltage level for the analog input that commands no motion. Valid range is 0.15 to 2.40 volts.

**DBL** Center deadband defines a range of voltage about the joystick center that commands no motion. Valid range is 0.00 to 1.24 volts.

**DB1** End deadband defines a voltage range at the upper and lower ends of the joystick input voltage beyond which the commanded velocity does not change. Valid range is 0.00 to 1.24 volts.

6000 command reference: JOYCTR, JOYCDB, JOYEDB

### Set Joystick Analog Inputs



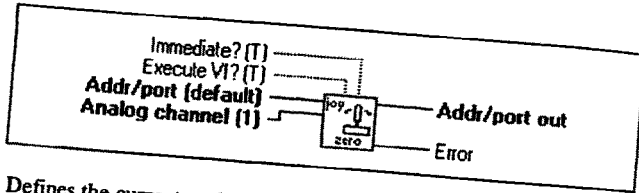
Specifies the analog channels that will control motion during joystick mode. The axes select input on the joystick connector determines which analog channel (low or high) will control the motion.

**IS2** Channel low defines the analog channel that will control the specified axis during joystick mode while the joystick axes select input is low.

**IS2** Channel high defines the analog channel that will control the specified axis during joystick mode while the joystick axes select input is high.

6000 command reference: JOYAXH, JOYAXL

## Set Joystick Zero



Defines the current analog value on the specified analog channel as joystick center. This VI will automatically determine center voltages, thus eliminating the need to use the Center input on the *Setup Joystick Electronics VI*.

**132** Analog Channel defines the specific analog channel to zero.

6000 command reference: JOYZ

## Miscellaneous VIs

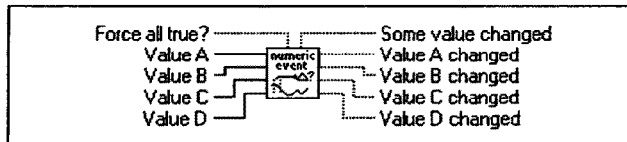
This chapter contains descriptions of the miscellaneous VIs included with Motion Toolbox. The figure below displays the *Miscellaneous* function palette.



Miscellaneous Function Palette

## Miscellaneous VI Descriptions

### Numeric Event



Monitors up to four numeric inputs for a change in value. A change in value flags an event that sets the corresponding output to true. The *Force all true?* input forces all outputs to be true. The *Some value changed* output indicates that one or more of the four inputs changed in value. The example below displays how the *Numeric Event* VI is used to only send velocities to the 6000 controller when the velocity setting changes.

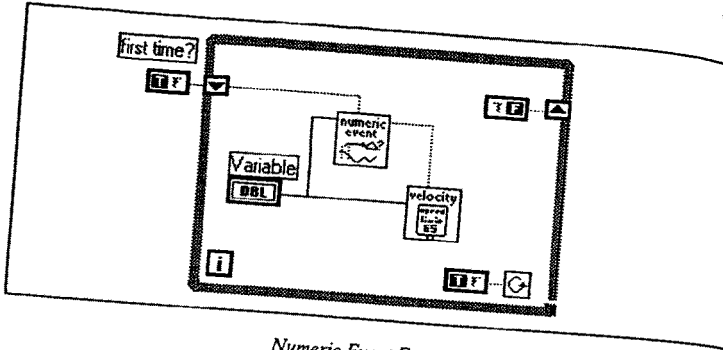
**DBL** Value A - D are the numeric values to monitor.

**TF** *Force all true?* When true, forces all outputs to be true. The example below displays how this input is used to force the *Set Velocity* VI to execute the first time through the while loop.

**TF** *Value A - D changed* is true if the corresponding input has changed in value.

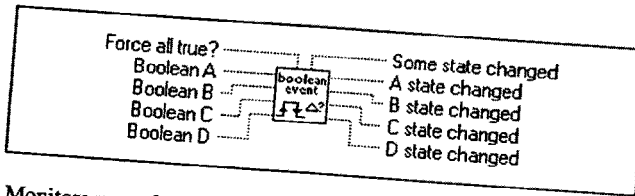
**TF** Some value changed is true if any of the four numeric inputs has changed in value.

6000 command reference: N/A



Numeric Event Example

### Boolean Event

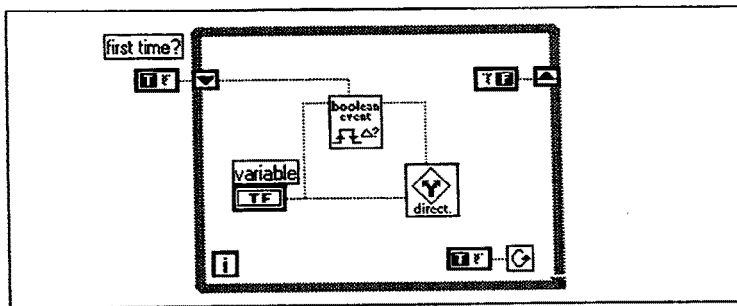


Monitors up to four boolean inputs for a change in state. A change in state flags an event that sets the corresponding output to true. The *Force all true?* input forces all outputs to be true. The *Some state changed* output indicates that one or more of the four inputs changed in state. The example below displays how the *Boolean Event VI* is used to only send direction information to the 6000 controller when the direction setting changes.

**DBL** Boolean A - D are the boolean values to monitor.

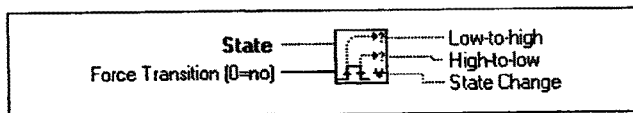
- TF** Force all true? When true, forces all outputs to be true. The Boolean Event example below displays how this input is used to force the *Set Direction* VI to execute the first time through the while loop.
- TF** A - D state changed is true if the corresponding boolean input has changed in state.
- TF** Some value changed is true if any of the four boolean inputs has changed in state.

6000 command reference: N/A



Boolean Event Example

### Boolean Transition



Monitors a boolean input for a state transition. A transition flags an event that sets the appropriate output to true.

- TF** State. The boolean variable to monitor.



**I32** **Force Transition.** Forces the state of this VI's outputs.

- 0: Do not force.
- 1: Force low-to-high transition.
- 2: Force high-to-low transition.

**TF** **Low-to-high.** Indicates monitored boolean input made a low-to-high transition.

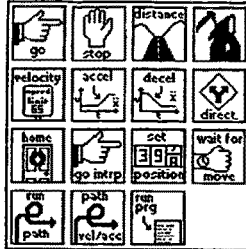
**TF** **High-to-low.** Indicates monitored boolean input made a high-to-low transition.

**TF** **State Change.** Indicates monitored boolean input made either a low-to-high or high-to-low transition.

6000 command reference: N/A

# Motion VIs

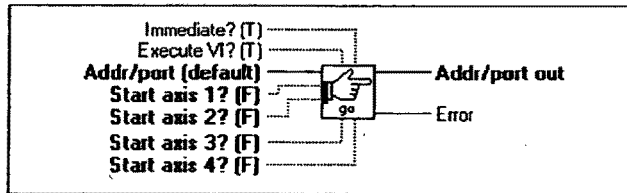
This chapter contains descriptions of the motion VIs included with Motion Toolbox. The figure below displays the *Motion* function palette.



Motion Function Palette

## Motion VI Descriptions

### Initiate Motion

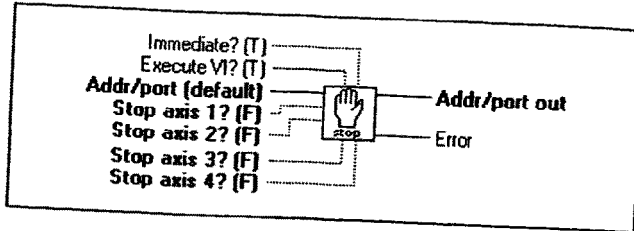


Initiates motion on the specified axes. If motion does not occur after executing this command, verify the drive fault level, pulse cutoff and limits are configured properly.

**TF** Start axis 1-4. A true initiates motion on the respective axis.

6000 command reference: GO

## Stop Motion

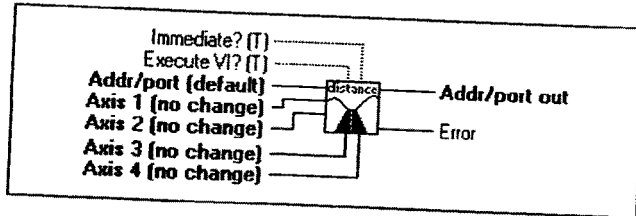


Stops motion on the specified axes. The deceleration used for stopping will be as set by the last deceleration command (see *Set Deceleration VI*).

**TF** Stop axis 1-4. A true stops the respective axis.

6000 command reference: S, COMEXS

## Set Distance

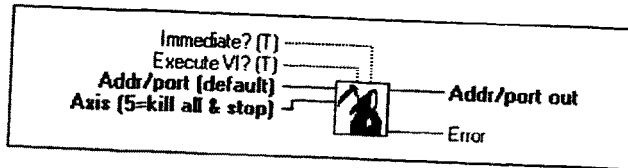


Defines either the number of units the motor will move or the absolute position it will seek after an *Initiate Motion* command. In incremental mode (see *Set Absolute/Incremental Mode VI*), the distance value represents the total number of units to move the motor. A positive distance indicates clockwise motion on the motor shaft (when viewed from the face of the motor), a negative value indicates counter-clockwise motion. In absolute mode, the distance value represents the absolute position at which motion will complete; the actual distance traversed will vary depending on the absolute position of the motor before the move is initiated.

**DBL** Axis 1 - 4 defines the distance for the respective axis.

6000 command reference: D

## Kill Motion



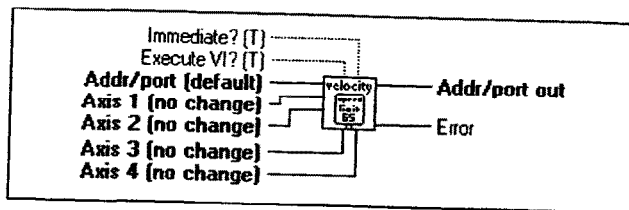
Instantaneously stops motion on the specified axes. This command is normally used in emergency situations with the Immediate input set to true allowing motion to stop upon the controller receiving the command.

**Warning for Stepper Systems:** This command should be used with caution. Since commanded motion is stopped instantaneously, without a controlled deceleration ramp, high inertial loads may cause a drive to fault. A drive fault condition will allow the load to free wheel, possibly damaging equipment. Compumotor recommends using a brake on your motor drive system to brake the load in the event of a drive fault. To have a controlled deceleration, use the *Stop Motion VI*.

- 152** **Axis.** Specifies the axis to kill.
- 0: Kills motion on all axes but continues command execution.
  - 1-4: Kills motion on specified axis.
  - 5: Kills motion on all axes and stops command execution.

**6000 command reference:** KILL, COMEXK

## Set Velocity

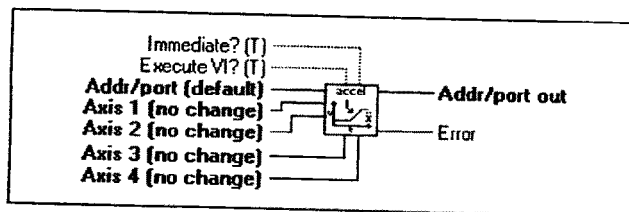


Sets the velocity for the specified axes in units per second. The velocity remains set until changed with a subsequent *Set Velocity* command. If scaling is enabled, the velocity values are internally multiplied by the velocity scaling factors (see *Set Motion Scaling Factors VI*).

**DBL** Axis 1-4 defines the velocity for the respective axis.

6000 command reference: V

## Set Acceleration

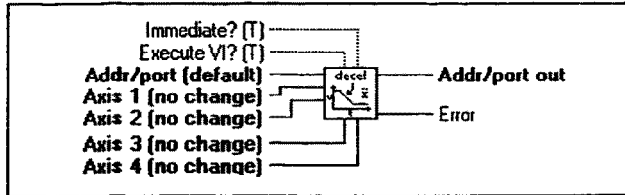


Sets the acceleration rate for the specified axes in units per second squared. The acceleration rate remains set until changed with a subsequent *Set Acceleration* command. If the deceleration rate has not been defined, the acceleration rate will be used to stop motion. Once the deceleration rate has been defined, the acceleration rate no longer affects deceleration. If scaling is enabled, the acceleration values are internally multiplied by the acceleration scaling factors (see *Set Motion Scaling Factors VI*).

**DBL** Axis 1 - 4 defines the acceleration for the respective axis.

6000 command reference: A

### Set Deceleration

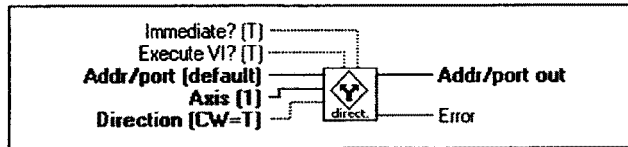


Sets the deceleration rate for the specified axes in units per second squared. The deceleration rate remains set until changed with a subsequent *Set Deceleration* command. If the deceleration rate has not been defined, the acceleration rate will be used to stop motion. Once the deceleration rate has been defined, the acceleration rate no longer affects deceleration. If scaling is enabled, the deceleration values are internally multiplied by the acceleration scaling factors (see *Set Motion Scaling Factors VI*).

**DBL** Axis 1-4 defines the deceleration for the respective axis.

6000 command reference: AD

### Set Direction



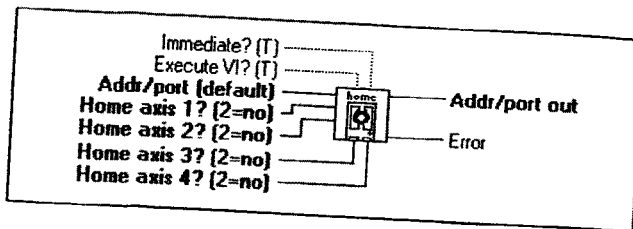
Sets the direction (clockwise or counter-clockwise) for the specified axis when in the incremental or continuous mode. The sign of a subsequent *Set Distance VI* will override this VI.

**TF** Direction.

- True: Set direction to clockwise.
- False: Set direction to counter-clockwise.

6000 command reference: D

## Go Home



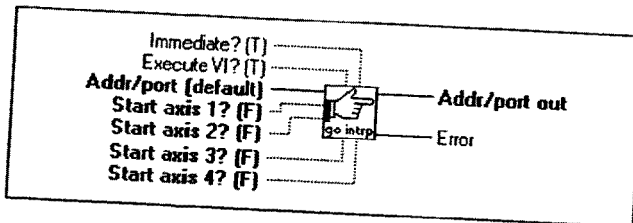
Instructs the controller to search for the home position in the direction, and on the axes specified. If an end-of-travel limit is activated while searching for the home limit, the controller will reverse direction and search for home in the opposite direction. However, if a second end-of-travel limit is encountered, after the change in direction, the homing operation is aborted. The status of the homing operation can be determined using the *Motion Status Parse VI*. When the homing operation is successfully completed, the absolute position register is set to zero.

**IS2** Axis 1-4 determines the direction of the homing search.

- 0: Home clockwise.
- 1: Home counter-clockwise.
- 2: Do not home.

6000 command reference: HOM

## Initiate Linear Int. Motion



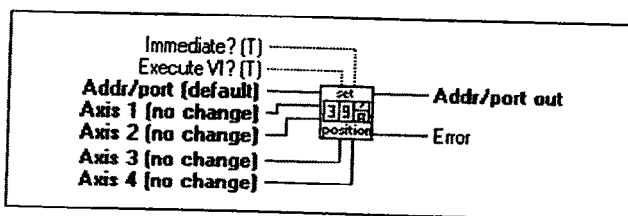
Initiates linear interpolated motion on the specified axes. Linear interpolated moves cause all axes to start and stop at the same time regardless of the distances traveled by each axis. The end point of the linear interpolated move

is dictated by the values entered using the *Set Distance VI*. The vector velocity and acceleration are determined by the *Set Path Velocity & Acceleration VI*. If motion does not occur after executing this command, verify the drive fault level, pulse cutoff and limits are configured properly.

**TF** Axis 1 - 4. A true initiates linear interpolated motion on the respective axis.

6000 command reference: GOL

### Set Position



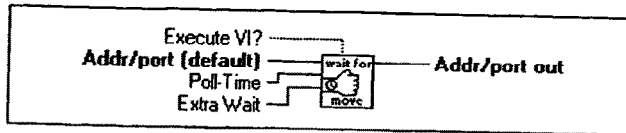
Defines the current, absolute-position counter for each axis. All values are in steps unless scaling is enabled (see *Enable Motion Scaling VI*), in which case values are multiplied by the distance scale factor (see *Set Motion Scaling Factors VI*).

**DBL** Axis 1 - 4 defines the absolute position for the respective axis.

6000 command reference: PSET



## Wait for Move Complete



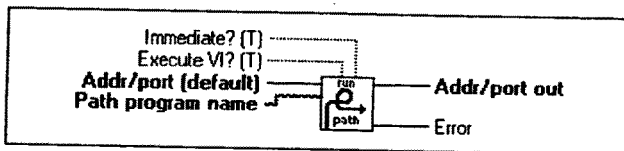
Waits for all motors to come to rest. This VI parses the *Misc. status* output of the *Get Fast Status* VI and waits until all motors are at rest before returning to its caller.

**U32** Poll-time specifies in milliseconds how often the axes are checked for a non-moving status.

**U32** Extra Wait specifies in milliseconds how long after motion has completed this VI waits before returning to its caller.

6000 command reference: N/A

## Run Path

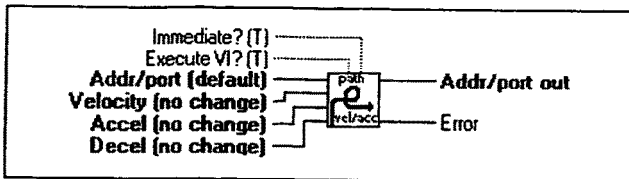


Executes a previously compiled (defined) path. If any of the axes included in the specified path are not ready, the path will not execute. An axis is not ready if it is in shutdown, moving, or in joystick or jog mode. When path execution begins, all included axes become busy until path execution is finished.

**abc** Path program name specifies the name of the path to execute.

6000 command reference: PRUN

### Set Path Vel & Acc



Specifies the velocity, acceleration, and deceleration values used in linearly interpolated and contouring moves. For both move types, the parameters refer to velocity, acceleration, and deceleration of the load as it proceeds along the path.

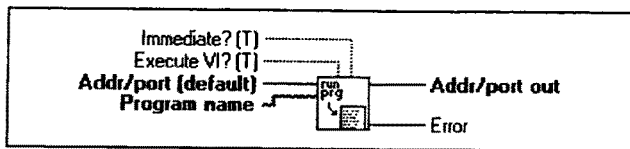
**DBL** Velocity defines the path velocity.

**DBL** Accel defines the path acceleration.

**DBL** Decel defines the path deceleration.

6000 command reference: PA, PAD, PV

### Run Program



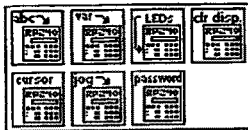
Executes a program previously defined with the DEF command. A program name must consist of 6 or fewer alpha-numeric characters. Use *Motion Architect* to define and debug your programs. You can then download your programs to the 6000 controller within LabVIEW using Motion Toolbox's *Download 6000 File VI*.

**abc** Program name specifies the name of the program to execute.

6000 command reference: RUN, DEF

## RP240 Display VIs

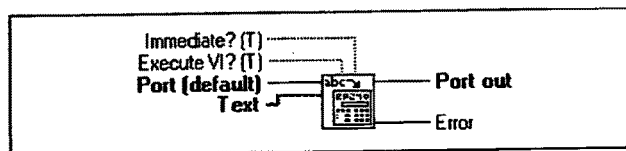
This chapter contains descriptions of the RP240 VIs included with Motion Toolbox. Only use the RP240 VIs with 62XX products. The figure below displays the *RP240 Display* function palette.



RP240 Display Function Palette

## RP240 VI Descriptions

### Write Text to RP240



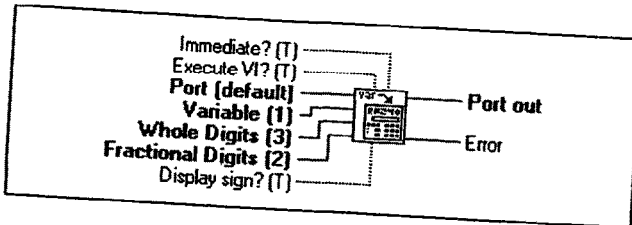
Writes the specified text to the RP240's display starting at the current cursor location. The message string may be up to 70 characters in length. The string may consist of any characters except quote ("), backslash (\), colon (:), and asterisk (\*).

Note: This VI is only applicable to the 62XX.

 Text to display.

6000 command reference: DWRITE

## Display Variable on RP240



Formats and writes the specified variable to the RP240's display starting at the current cursor location.

Note: This VI is only applicable to the 62XX.

**I32** Variable to display. Valid range is 1 to 150.

**I32** Whole Digits to display. Valid range is 0 to 9.

**I32** Fractional Digits to display. Valid range is 0 to 8.

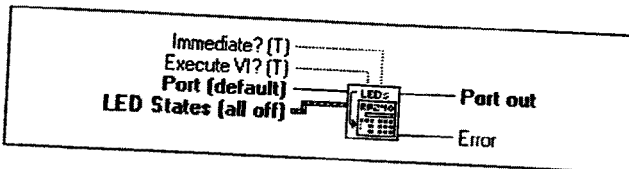
**TF** Display sign? determines if the + or - sign is displayed.

True: Display sign.

False: Do not display sign.

6000 command reference: DVAR

## Set RP240 LED States



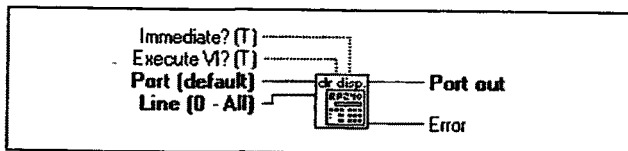
Controls the state of the eight programmable LEDs on the RP240.

Note: This VI is only applicable to the 62XX.

**598** LED States is a cluster of 8 booleans. Each boolean controls the corresponding LED on the RP240.

6000 command reference: DLED

### Clear RP240 Display



Clears the specified line(s) of the RP240's display.

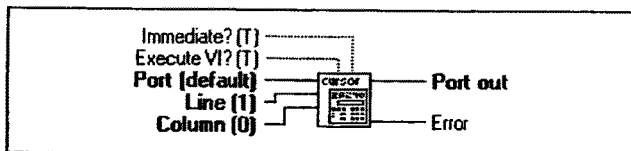
Note: This VI is only applicable to the 62XX.

**152** Line to clear.

- 0: Clear all lines.
- 1-2: Clear specified line.

6000 command reference: DCLEAR

### Position RP240 Cursor



Changes the location of the cursor on the RP240 display.

Note: This VI is only applicable to the 62XX.

**IS2** Line to position cursor.

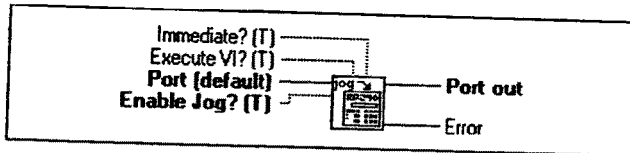
- 0: Position to top line.
- 1: Position to bottom line.

**IS2** Column to position cursor. Column 0 is on the far left and 39 is on the far right.

0-39: Column position.

6000 command reference: DPCUR

### Enable RP240 Jog Mode



Enables the RP240 jog mode on all axes. Once enabled, the operator can use the RP240 arrow keys to jog individual axes. Note: The 6200 suspends processing after enabling jog mode.

The jogging VIs for acceleration, deceleration, and velocity can be used with the RP240 jog mode. Once in this mode, the operator can switch between low and high jog velocities for any axis, and can also modify the two jog velocities using the RP240's EDIT key.

To disable the RP240 jog mode, press the MENU RECALL key or call this VI with *Enable Jog?* set to false. Upon exiting the RP240 jog mode, the RP240's display is cleared.

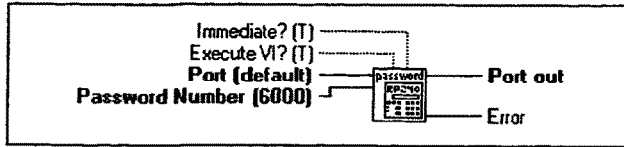
Note: This VI is only applicable to the 62XX.

**TF** Enable Jog?

- True: Enable RP240 Jog mode.
- False: Disable RP240 jog mode.

6000 command reference: DJOG

## Set RP240 Password



Sets the RP240 password. If the 62XX default password is not changed then the RP240 is not password protected.

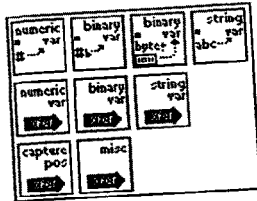
Note: This VI is only applicable to the 62XX.

**IS2** Password number. A number in the range of 1-9999 that defines the RP240 password.

6000 command reference: DPASS

# Variable & Transfer VIs

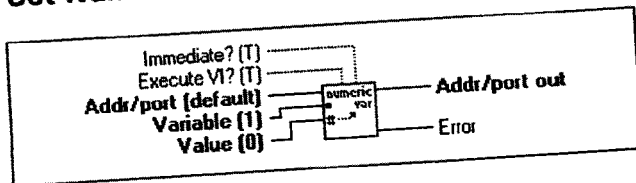
This chapter contains descriptions of the variable and transfer VIs included with Motion Toolbox. The figure below displays the *Variable & Transfer* function palette.



Variable & Transfer Function Palette

## Variable & Transfer VI Descriptions

### Set Numeric Variable



Sets the value of a numeric variable. To retrieve the value of a numeric variable, use the *Transfer Numeric Variable VI*.

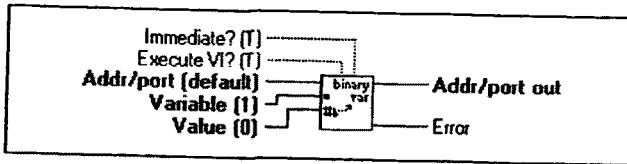
**IS2** Variable. Specifies the variable to set. Valid range is 1 to 100.

**DB1** Value is a real number in the range of -999,999,999.99999 to 999,999,999.99999.

6000 command reference: VAR



## Set Binary Variable



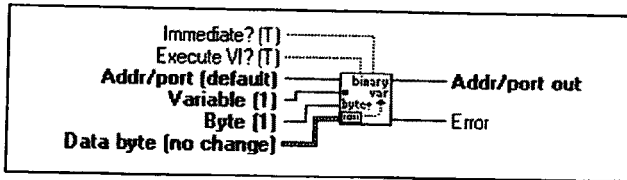
Sets the value of a binary variable. To retrieve the value of a binary variable, use the *Transfer Binary Variable VI*.

**IS2** Variable. Specifies the variable to set. Valid range is 1 to 25.

**IS2** Value is a 32-bit binary number.

6000 command reference: VARB

## Set Binary Variable by Byte



Sets the value of a binary variable by byte. To retrieve the value of a binary variable, use the *Transfer Binary Variable VI*.

**IS2** Variable. Specifies the variable to set. Valid range is 1 to 25.

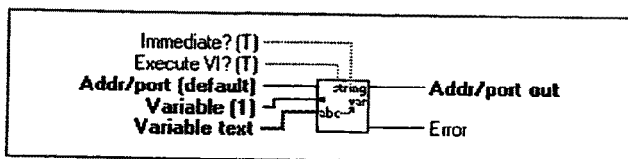
**IS2** Byte specifies which of the four bytes of a 32-bit binary number to set. Valid range is 1 - 4 where 1 corresponds to the least significant byte.

**508** Data byte is a cluster of eight 32-bit integers that represent the desired state for each bit of the data byte.

- 0: Turn bit off.
- 1: Turn bit on.
- 2: Do not change current state.

6000 command reference: VARB

### Set String Variable



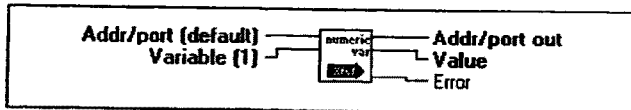
Sets the contents of a string variable. To retrieve the contents of a string variable, use the *Transfer String Variable VI*.

**132** Variable. Specifies the variable to set. Valid range is 1 to 25.

**132** Variable text is the text to assign to the variable. Strings can be up to 20 characters in length. Do not include quotes ("), semicolons (;) or colons (:) in the string.

6000 command reference: VARS

### Transfer Numeric Variable



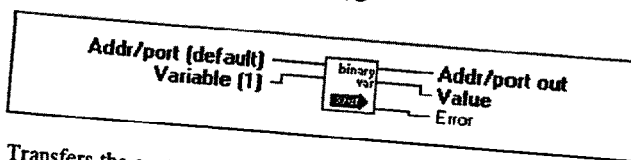
Transfers the contents of a numeric variable from the 6000 controller. To set the contents of a numeric variable, use the *Set Numeric Variable VI*.

**I32** Variable to transfer. Valid range 1 to 100.

**U32** Value transferred.

6000 command reference: VAR

### Transfer Binary Variable



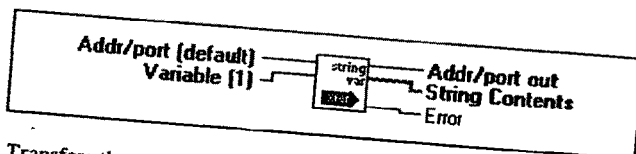
Transfers the contents of a binary variable from the 6000 controller. To set the contents of a binary variable, use the *Set Binary Variable* or *Set Binary Variable by Byte* VIs.

**I32** Variable to transfer. Valid range is 1 to 25.

**U32** Value transferred.

6000 command reference: VARB

### Transfer String Variable



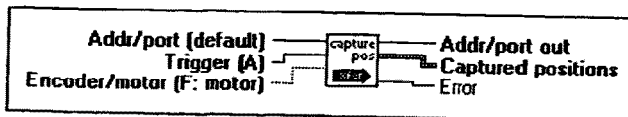
Transfers the contents of a string variable from the 6000 controller. To set the contents of a string variable, use the *Set String Variable* VI.

**I32** Variable to transfer. Valid range 1 to 25.

**abc** String Contents of the variable transferred.

6000 command reference: VARS

## Transfer Captured Positions



Transfers the current captured encoder or motor positions, from the time of the last trigger interrupt. After transferring, the respective position capture status bit (See *Pos. Captured Status Parse VI*) is cleared, but the position information remains in the register until it is over written by a subsequent position capture from the trigger input. To configure inputs to capture positions, use the *Set 6000 Input Function VI*.



**Trigger.**

- 0: Transfer captured positions corresponding to the A trigger.
- 1: Transfer captured positions corresponding to the B trigger.
- 2: Transfer captured positions corresponding to the C trigger.
- 3: Transfer captured positions corresponding to the D trigger.



**Encoder/motor.**

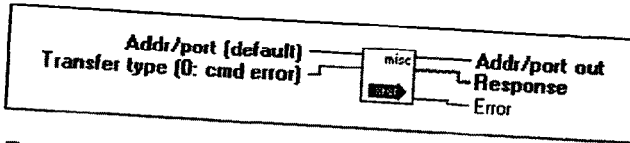
- True: Transfer encoder positions for trigger.
- False: Transfer motor positions for trigger.



**Captured positions.** A cluster of four, 32-bit integers indicating the transferred captured positions for axes 1 -4 respectively.

6000 command reference: TPCE, TPCM

## Miscellaneous Transfers



Transfers miscellaneous status information from the 6000 controller.

**132** **Transfer type**

- 0: Transfer the command, if any, the controller has detected as an error.
- 1: Transfer software revision information.
- 2: Transfer memory usage status.

**abc** **Response.** The transferred response to the specified query.

6000 command reference: TCMDER, TREV, TMEM

## Appendix

---

### Motion Toolbox Error Codes

Error	Description
1	Get DLL procedure address failed. Verify WIN6400.DLL version is compatible with this version of Motion Toolbox.
2	Get indirect handle failed. Verify WIN6400.DLL version is compatible with this version of Motion Toolbox.
3	Unable to load DLL. Ensure WIN6400.DLL is in LabVIEW root directory.
4	AT6400 Operating System not loaded. Load AT6400 Operating System before executing any other Motion Toolbox VIs.
5	Timeout while communicating with AT6400. Ensure AT6400 is in computer and address specified is correct. Verify 6000 device functions properly with Compumotor's Motion Architect.
6	Load AT6400 operating system failed. Verify AT6400 is in computer and address specified is correct. Verify board functions properly with Compumotor's Motion Architect.
7	Command string exceeds 256 characters.
8	Failure while parsing query response.
9	Not used.
10	Command not compatible with device.
11	Receive from 6000 exceeded user specified timeout period.
12	Parsing error getting fast status from 6200. Ensure RS-232 communications is functioning properly.

## **Technical Support**

For technical support, please call the Compumotor Applications Department at 800-358-9070 or 707-584-7558. You can also send a fax to 707-584-3793. Please be prepared to state your product serial number when calling in for assistance.

**Parker Compumotor**  
5500 Business Park Drive  
Rohnert Park, CA 94928

## **Compumotor Bulletin Board Service**

Compumotor offers a bulletin board service (BBS) -- free of charge. The BBS allows you to send or receive messages and download example programs that demonstrate the use of Motion Toolbox, 6000 Series controllers, and other products.

To dial in, you must have at least a 2400 baud modem with your computer. Set the communications parameters to 8 data bits, 1 stop bit, and NO parity; virtually all communications programs will allow you to set these parameters. The BBS number is 707-584-4059.

Once connected, you will be asked some questions about yourself. When you have completed the personal information, you are free to explore the services of the bulletin board.

## **Systems Integration and Consulting**

Motion Toolbox systems integration and consulting services are available through Snider Consultants, Inc. Motion Toolbox was developed by Snider Consultants, Inc. through an alliance with Parker Compumotor. Snider Consultants, Inc. is a full-service systems integration and consulting firm specializing in the development of LabVIEW-based systems.

**Snider Consultants, Inc.**  
5100-202 Tybrook Court  
Raleigh, NC 27612  
(919) 787-8008  
Fax (919) 571-7985

## Motion Toolbox VI Index (alphabetical)

Analog Input Parse .....	54	Initiate Motion .....	85
Boolean Event .....	82	Joystick Status Parse .....	55
Boolean Transition .....	83	Kill Motion .....	87
Clear RP240 Display .....	97	Limit Input Status Parse .....	54
Close Device .....	33	Miscellaneous Transfers .....	106
Command Error Parse .....	47	Motion Status Parse .....	49
Command Snooper .....	40	Motor Position Parse .....	45
Configuration Status Parse .....	48	Numeric Event .....	81
Configure Command Control .....	18	Open 6200 .....	32
Configure Communications Tracing .....	39	Open AT6400 .....	31
Configure Encoder Input as Counter .....	12	Other Input Parse .....	56
Configure Feedrate Override .....	22	Pos. Captured Status Parse .....	47
Configure Hard Limits .....	67	Position RP240 Cursor .....	97
Configure Homing .....	24	Query 6000 .....	37
Configure Position Maintenance .....	26	Receive 6000 Block .....	37
Configure Soft Limits .....	68	Reset 6000 .....	33
Configure Stall Detection .....	27	Reset 6000 Hardware Counter .....	13
Define 6000 User Status .....	71	Run Path .....	92
Delete Trace File .....	40	Run Program .....	93
Digital Input Parse .....	57	Send 6000 Block .....	36
Digital Output Status Parse .....	56	Set 6000 Input Active Level .....	59
Display Variable on RP240 .....	96	Set 6000 Input Debounce Time .....	62
Download 6000 File .....	38	Set 6000 Input Function .....	61
Drive Status Parse .....	52	Set 6000 Output Active Level .....	63
Enable 6000 Input Functions .....	62	Set 6000 Output Function .....	64
Enable 6000 Inputs .....	60	Set 6000 Output States .....	66
Enable 6000 Output Functions .....	65	Set 6200 Polling Parameters .....	36
Enable 6000 Outputs .....	64	Set Absolute/Incremental Mode .....	19
Enable 6000 User Status .....	72	Set Acceleration .....	88
Enable Analog Input Override .....	69	Set Analog Input Override Voltage .....	70
Enable Communications Tracing .....	38	Set AT6400 Polling Parameters .....	35
Enable Drive .....	21	Set Binary Variable .....	102
Enable Feedrate Override .....	23	Set Binary Variable by Byte .....	102
Enable Hard Limits .....	67	Set Continuous/Preset Mode .....	19
Enable Jog Mode .....	73	Set Deceleration .....	89
Enable Joystick Mode .....	76	Set Default Addr/Port .....	33
Enable Position Maintenance .....	27	Set Direction .....	89
Enable RP240 Jog Mode .....	98	Set Drive Fault Level .....	20
Enable Scale Factors .....	17	Set Drive Resolution .....	20
Enable Soft Limits .....	69	Set Encoder Pos Local Scaling .....	43
Enable Stall Detection .....	28	Set Encoder Resolution .....	23
Encoder Feedback Status Parse .....	48	Set Encoder/Motor Step Mode .....	24
Encoder Position Parse .....	45	Set Error Action .....	34
Get Fast Status .....	41	Set Jog Acceleration .....	75
Go Home .....	90	Set Jog Deceleration .....	75
Hard Limit Status Parse .....	49	Set Jog Velocity High .....	74
Initiate Linear Int. Motion .....	90	Set Jog Velocity Low .....	74



*Motion Toolbox User Guide*

Set Joystick Acceleration .....	77
Set Joystick Analog Inputs.....	79
Set Joystick Deceleration.....	78
Set Joystick Velocity High .....	77
Set Joystick Velocity Low.....	76
Set Joystick Zero.....	80
Set Motion Scaling Factors .....	15
Set Motor Pos Local Scaling.....	43
Set Numeric Variable.....	101
Set Parameter Precision .....	34
Set Participating Axes .....	17
Set Path Scaling Factors.....	16
Set Path Vel & Acc.....	93
Set Position.....	91
Set Pulse Width.....	29
Set RP240 LED States.....	96
Set RP240 Password.....	99
Set String Variable.....	103
Set Velocity .....	88
Set Velocity Local Scaling.....	44
Setup Joystick Electronics .....	78
Soft Limit Status Parse .....	50
Start 6000 Timer .....	11
Stop 6000 Timer .....	12
Stop Motion .....	86
System Status Parse .....	51
Timer Status Parse .....	53
Transfer Binary Variable.....	104
Transfer Captured Positions .....	105
Transfer Numeric Variable .....	103
Transfer String Variable .....	104
User Status Parse .....	53
Velocity Parse .....	46
Wait for Move Complete.....	92
Write Text to RP240 .....	95