Remote Interface Guide

Velox

PN 162-043, Revision 15, June 14, 2019

FormFactor™

© Copyright 2019 FormFactor, Inc. All rights reserved. No part of this document may be reproduced, transmitted or displayed in any form or by any means except as duly authorized by FormFactor, Inc. FormFactor and the FormFactor logo are trademarks of FormFactor, Inc. All other trademarks are the property of their respective owners.

Important Notice

While the information contained herein is believed to be accurate as of the date hereof, no express or implied representations or warranties are made with respect to its accuracy or completeness. FormFactor, Inc., and its subsidiaries disclaim liability for any inaccuracies or omissions. All information is subject to change without notice.

Users are required to read and follow carefully all safety, compliance and use instructions. Users assume all loss and liability arising from the use of products in any manner not expressly authorized. The conditions and methods of use of products and information referred to herein are the entire responsibility of the user and, to the maximum extent permitted by applicable law, FormFactor, Inc., and its subsidiaries shall not be liable for any damages, losses, costs or expenses arising out of, or related to, the use thereof.

No license, express or implied, by estoppel or otherwise, under any intellectual property right is granted in connection herewith. Users shall take all actions required to avoid intellectual property infringement.

Contents

Welcome ..... **vi**
- Notational Conventions vi
- For More Information vii
- Technical Support vii

Chapter 1: System Control Interface **1**
- SCI/PC Configuration 2
- Coordinate Systems 3
  - Deviations from Normal Orientation 3

Chapter 2: RS232 Interface **5**
- Command Protocol 5
- User Interface 7
- Settings 8

Chapter 3: TTL Interface **9**
- Protocol 9
  - Standard Test Procedure 9
    - Binning Order During Cluster Measurements 10
  - TTL Cable Pinout 11
- User Interface 11
  - Current State 12
    - Current Tasks 12
    - TTL Interface Tasks 13
  - Settings 14
  - Advanced Settings 14
  - Test Signals 15

Chapter 4: GPIB Interface **17**
- Overview 17
  - Message Server Command Processing 17
- Protocol 17
  - Termination 18
  - Command Protocol for SCI type commands 18
  - Serial Polling 20
- GPIB Commands 20
- Asynchronous GPIB communications 24
  - Example Sequences 25
- User Interface 25
  - Board Configuration 27

Chapter 5: Message Server **29**
User Interface 30
Toolbar 30
Applications 30
Message Log 31
Log Filter Configuration 31
Connecting to the Message Server 32
Chapter 6: Scripting Console **36**
Scripting Examples 37
Simple Sequential Scripting 37
Python Scripting 38
User Interface 40
Toolbars 40
Menus 41
File Menu 41
Edit Menu 42
Commands Menu 43
Run/Debug Menu 43
Options Menu 44
Help Menu 45
Status Bar 45
Key Shortcuts Quick Reference 46
Script Editor Window 46
Editor Pages 47
Editor Breakpoints and Line Numbers 48
Command Chooser 48
Output Windows 50
Simple Sequential Scripting Language 51
Simple Sequential Scripting 51
Python Scripting Language 52
Python Overview 53
Commands 55
SCI Commands 55
Miscellaneous Scripting Commands 57
User-Defined SCI Commands in Python Script 58
Symbol Navigation 59
Script Output 59
IntelliSense and Automatic Code Completion 59
Introduction to Python Programming 60
Imported Legacy Remote Communicator Scripts 61
File Conversion 62

Simple Script .SCR Files 62
Basic Script .BAS Files 62
File Operations 62
Optimizing Imported Scripts 62
Comment Lines 63
Library Functions and Limitations 63
GPIB Serial Communications 64
Configuring the Interfaces 64
Configuring GPIB 64
Configuring Serial Ports 64
Error Handling 64
WinCal Remoting with Scripting Console and Python Script 65
Using WinCal Remoting from Scripting Console 65
Importing WinCal Remoting Modules 65
Connecting to the WinCal Server 65
Calling WinCal APIs From Python Script 65
Chapter 7: Server Commands 67
Common Commands 67
Notifications 72
Chapter 8: Application Commands 76
Optical Control Commands 76
Scripting Console Commands 80
Spectrum Commands 81
Camera Commands 81
General Commands 84
ReAlign Commands 88
Tool Commands 91
TTL Interface Commands 100
VeloxPro Commands 101
Wafer Map Commands 110
Map Commands 110
Die Commands 126
Subdie Commands 136
Cluster Commands 146
Bin Commands 151
Profiling Commands 154
Chapter 9: Kernel Commands 157
Alignment Commands 157
Chuck Commands 161
Positioner Commands 174
Scope Commands 190

AUX Site Commands 199
Theta Commands 209
Turret Commands 211
System Commands 212
Offset Commands 226
Setup Commands 227
Temperature Chuck Commands 229
Notifications 234
ASCII Strings in Notifications 238

Chapter 10: Error Codes 240
Kernel Errors 240
B&R Controller Errors 244
Server and Common Errors 245
Application Errors 246

Appendix A: Legacy Commands: SCPI 250
Summary of Commands 250
SCPI Device IDs 256
SCPI Descriptions 256
Alignment Commands 256
Automation Commands 258
Capture Video Command 260
Move Commands 261
Wafer Map Commands 292
Reverse Orientation Command 313
Set Commands 314
System Commands 342
Thermal Chuck Commands 347
Vision Commands 350

Appendix B: Legacy Remote Communicator 353
User Interface 353
Toolbar 354
Log Window 354
Script Pane 355
Quick Command Window 356
Status Bar 356
File Menu 357
Edit Menu 357
Commands Menu 357
Command Groupings 358
Action Menu 359

Setup Menu 360
Configuration Interfaces 360
Configuration Logging 361
Configuration Miscellaneous 362
Notifications Handling Script 363
View Menu 363
Switch to MiniCommunicator 363
Help Menu 364
Typical Setup 364
Keyboard Functions 365
Script Languages 366
Simple Script Language 366
Script Language (Pascal/Basic) 366
General Script Description (Pascal/Basic) 367
Special Script Description 373
Internal Function Descriptions 375
Appendix C: Legacy Communication Protocols 403
DDE Server Communications 403
DDE Programming Examples 405
Index 408

Welcome

Welcome to the Velox Remote Interface Guide. The information provided in this guide describes the use of Velox software for automated probe station operation.

NOTE

Only FormFactor supplied controllers are supported. Adding third-party software or drivers, or modifying the software installed by FormFactor, may void your warranty.

Notational Conventions

This manual uses the following conventions:

- Commands are not case-sensitive.
- Parameters are case-sensitive in Python.
- Syntax strings (commands) appear in this font:

  `GetDieMapResult1 1 2`

NOTE

Note is used to indicate important information about the product that is not hazard related.

CAUTION

Caution is used to indicate the presence of a hazard which will or can cause minor personal injury or property damage if the warning is ignored.

WARNING

Warning is used to indicate the presence of a hazard which can cause substantial personal injury or property damage if the warning is ignored.

DANGER

Danger is used to indicate the presence of a hazard which will cause severe personal injury, death or substantial property damage if the warning is ignored.

For More Information

More information is available from these sources:

- Velox Help
- Release Notes on the Windows Start menu > FormFactor Documentation
- www.formfactor.com

Technical Support

If it is necessary to contact customer service, you can use the Error Report tool to provide the appropriate information. See the Velox Help for details.

For immediate sales support or customer service assistance, please contact us at 1-800-550-3279 or 1-503-601-1000.

Chapter

1 System Control Interface

The System Control Interface (SCI) provides a remote interface to applications running on the same PC as Velox as well as on other PCs or embedded controllers. All applications used to control the prober use this interface The internal communications are handled with a Windows dynamic link library (DLL).

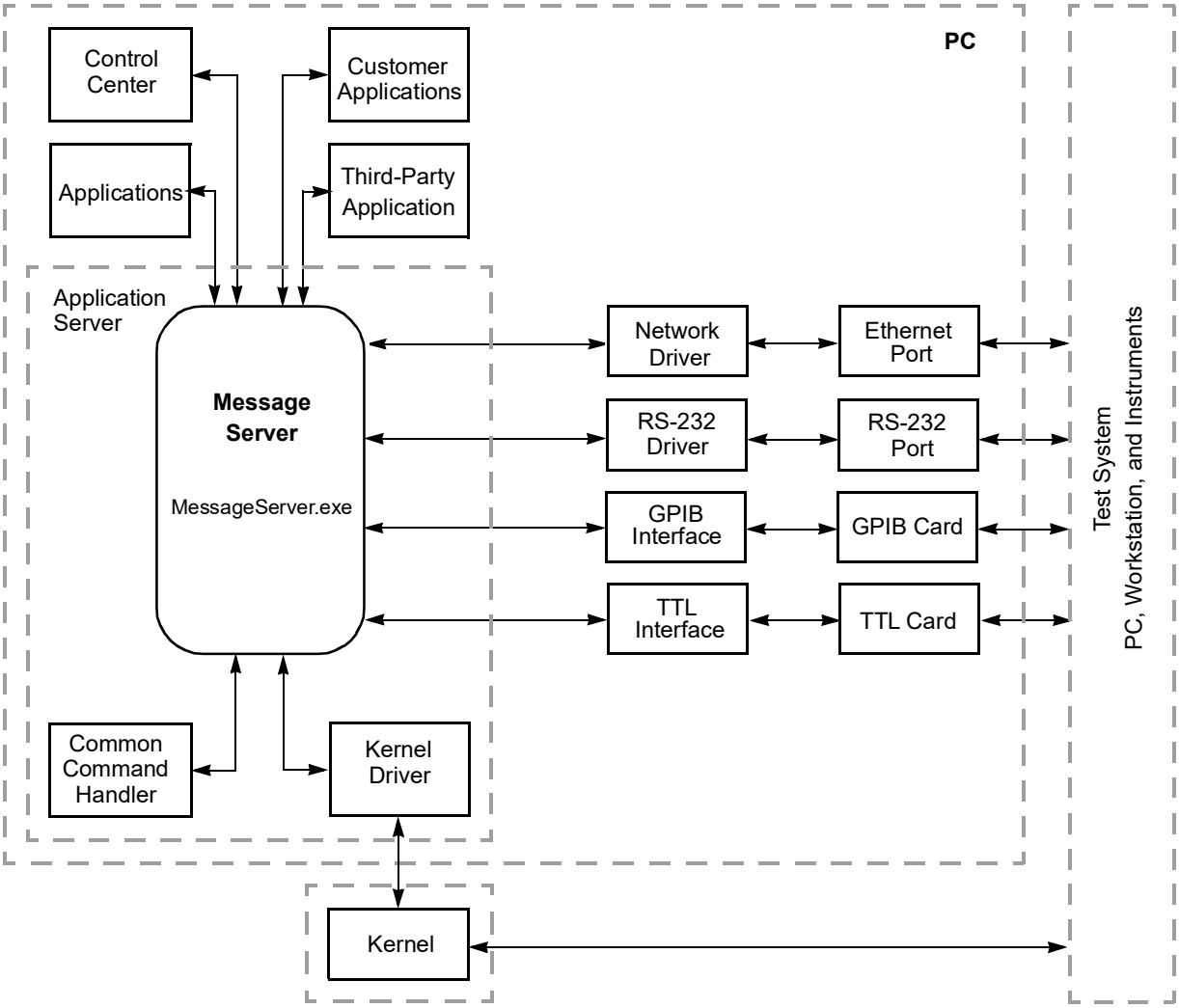
	The GPIB Interface provides system communication integration using a General Purpose Interface Bus (GPIB) conforming to the IEEE-488 bus protocol.	Access GPIB, RS232, or TTL from the main Velox menu (Velox > All applications > Interfaces) or the Velox Tools folder in the Start menu.
	The RS232 Interface provides system communications using a RS232 serial port, common to most PCs. No additional hardware is required other than a free RS232 port.	
	The TTL Interface provides communications between probe systems and testers which use TTL-type communications.	
	The Message Server is the central application that dispatches messages between the other Velox applications. During Velox operation, Message Server runs in the background.	To open Message Server, click the Show hidden icons button in the task bar.
	The Scripting Console can be used by programmers to view SCI syntax, send commands, and read prober responses, as well as by Service to test individual components of the system.	Access Scripting Console from the Velox toolbar and from the Start menu.

CAUTION

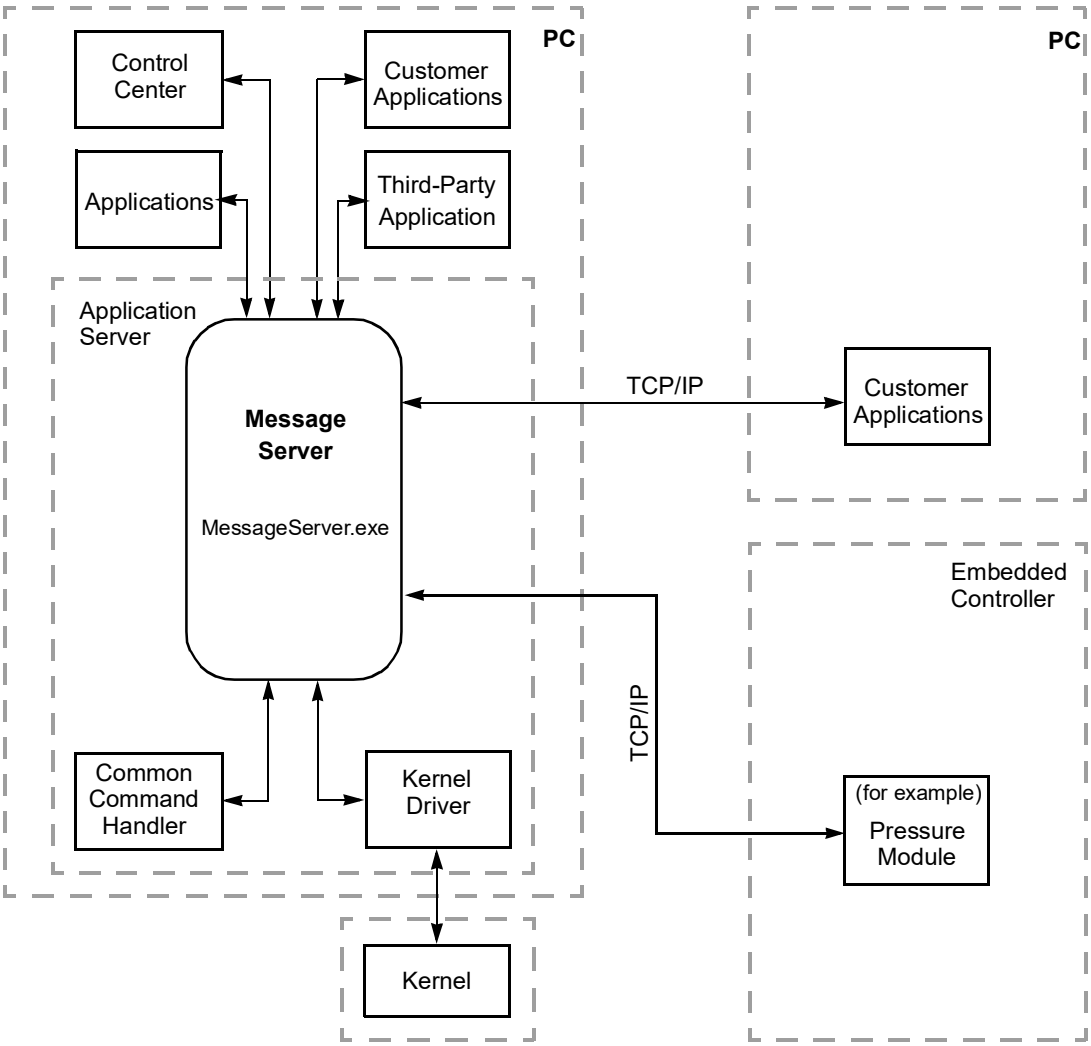
If the operations are disrupted by resetting the PC or Message Server, settings such as Contact Height may be lost. Make sure to check all settings before proceeding.

SCI/PC Configuration

Applications running on a single PC



Applications running on multiple PCs



Coordinate Systems

The basic coordinate system rules are:

Positive counting direction	Direction to the wafer contact
Safe direction	Move to zero position

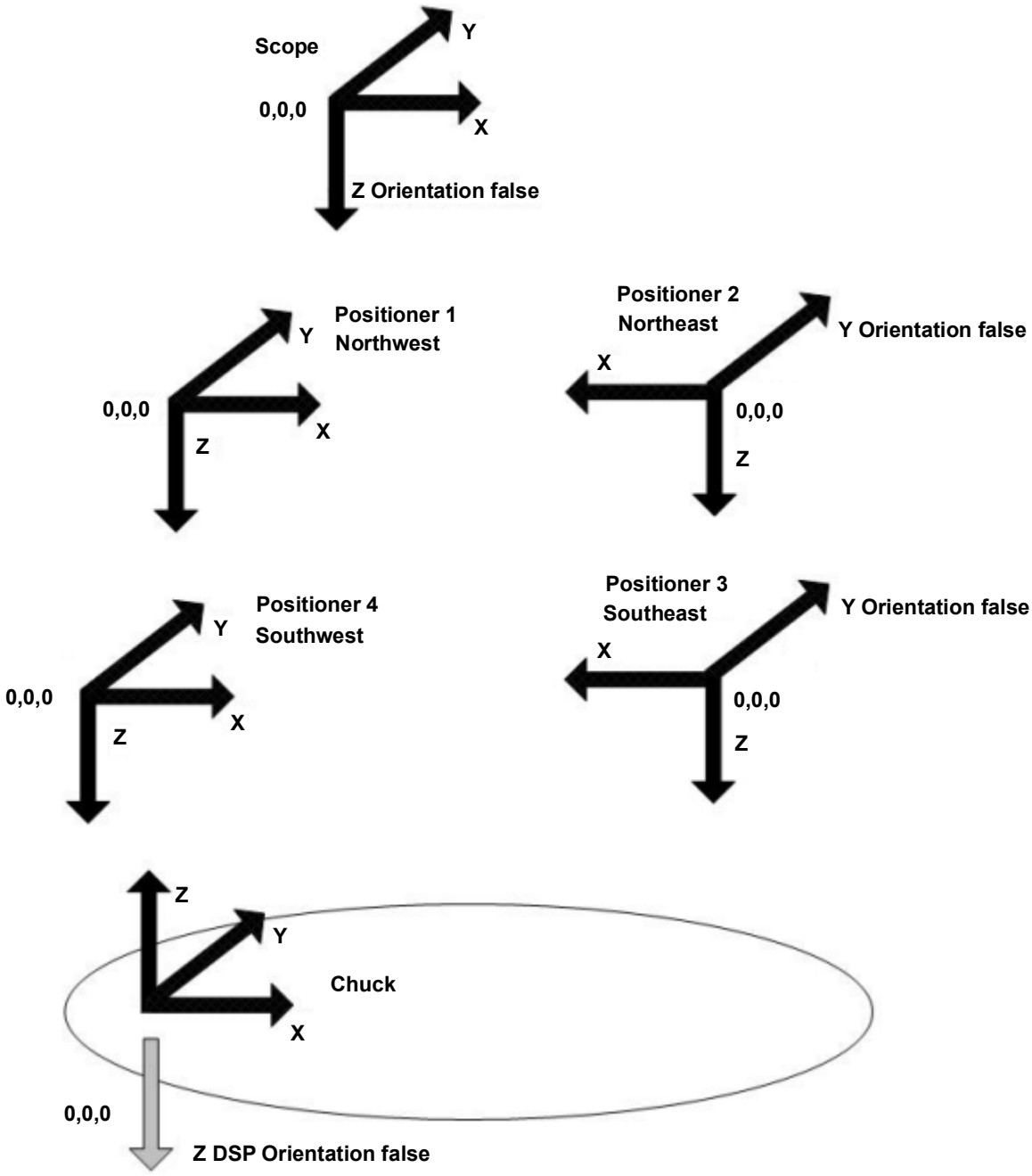
Deviations from Normal Orientation

Sometimes logical turns of X, Y or Z axes are necessary, deviating from normal orientation. These logical turns are implemented in the Kernel software.

Scope	Z orientation
Positioners	Y orientation, depending on which side of the platen they are placed
Chuck	Z orientation, in case of DSP prober (backside mode)

NOTE

The direction to the wafer contact is the positive counting direction.



Chapter

2 RS232 Interface

Click on the

icon on the Velox toolbar or use the shortcut in the Velox folder.

A common source of RS232 communication problems is inconsistent termination settings between the RS232Interface.exe and the master (command sender) application.

The RS232Interface.exe Menu Configuration\Interface Driver\Terminator sets up the Terminator. Values include:

- Carriage Return (CR),
- Carriage Return/Line Feed (CR/LF)
- Line Feed

Be sure, that RS232Interface.exe and the Master (Sender) Application uses the same Terminator.

- 10 (ASCII Line Feed)
- 13 (ASCII Carriage Return)

Command Protocol

Normal Command String
Normal Response String
Post Command Response String
Response Separator
Error Response String

All commands and responses are ASCII text strings. All commands and responses, and the respective data arguments, are separated by spaces. Commands are not case sensitive. If RS232 gets a terminated (command) string, it interprets the string as Velox Remote Command.

Every Remote Command returns a Response string, starting with an error code, even if it does not contain any additional response parameters.

Commands can either be sent synchronously (Normal) or as a post command.

Normal (synchronous)	The response string is returned after the command is completed.
Post command	The response returns immediately, but does not have any further information.

If a synchronous sent command is pending, and RS232 gets another command, it is not sent to MsgServer, until the pending command is done.

Normal Command String

A command string consists of a command name followed by a space, then followed by any data necessary, separated by spaces. The exact data depends on the command. See the remote command set definition for a list of commands that may be sent. Data is either decimal (float) or

whole (char, Unit, Int) numbers. The command ID number can be sent instead of the command name.

Description

Moves the chuck to the specified position.

Example

```
34 1233.5 -67745.3
```

Normal Response String

If no error occurs during the command execution a normal response string is generated from the prober, if requested. The response normally consists of a zero followed by an error separator Remote_Error_Separator character. If the command executed returned data (for example, ReadChuckPosition) this data will appear after the error separator.

Description

Returns the actual chuck position in x, y and z.

Example

```
ReadChuckPosition
0 123.4 -567.8 1022.8
```

Post Command Response String

The post command response returns immediately (before the command is done). The response is the command itself and does not contain further information.

Description

Returns the post command response.

Example

```
MoveChuckIndex 1 0 R Y 100
0 <MoveChuckIndex
```

Response Separator

The error separator is the character that appears between the 0 and the returned data in a command response and between the error number, the error text and the original command in an error response.

Description

Returns the post command response.

Example

```
ReadChuckPosition
0 0.5 1.8 14.7
```

```
StepProbeSite 1 13
82 X Axis on End Limit:StepProbeSite 1 13
```

Error Response String

The response consists of an error# (a whole number between 1 and 999) followed by an error separator Remote_Error_Separator, followed by the error source and the error text.

Description

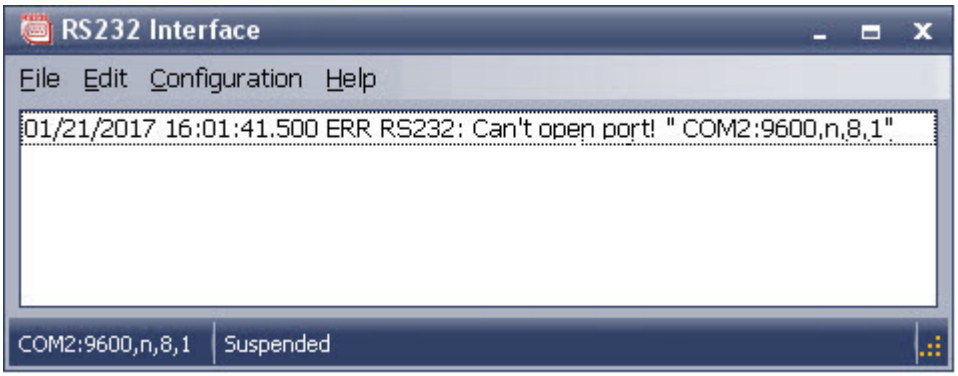
Error# 75 on a expected software fence violation after the commanded chuck movement.

Example

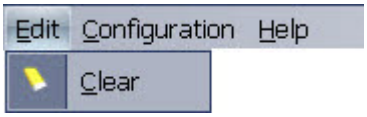
```
MoveChuckIndex 1 0 R Y 100
75 [Chuck] Software fence error
```

User Interface

The user interface contains the controls to configure the RS232 interface and displays the port activity.



Menus

	• Shows or hides the RS232 interface • Shuts down the RS232 interface
	• Clears the activity pane
	• Interface Driver opens the Port Settings window to change port configuration • Reset Port closes, then reopens the communications port

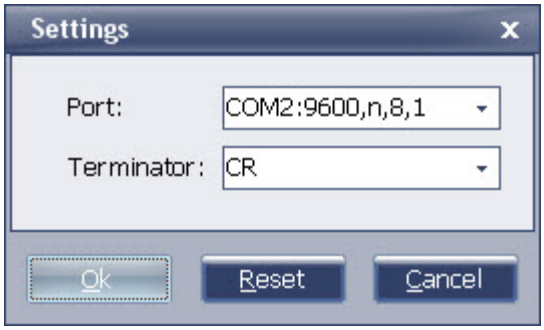
- Opens the Help file
- Displays Velox application, copyright and version information

Settings

Multiple RS232 ports may be used simultaneously. The configuration of the first port will be read from the configuration file. Subsequent ports must be passed to the RS232 application as command line parameters. For example: RS232 COM4:19200,n,8,1.

When the port is configured from the command line, the options cannot be saved to the configuration file.

Configuring ports from the command line will not prevent the application from reading other settings from the configuration file at start up.



Port

Select the port from the drop box. Example: COM1:9600,n,8,1

Port number	COM1
Bits per second	9600
Parity	n
Data bits	8
Stop bits	1

Terminator

The drop-down list allows you to select how commands should be terminated. Be sure that the software sending the commands is set to the same option.

CR (default)	Carriage Return – Decimal code is 13
LF	Line Feed – Decimal code is 10.
CR/LF	Combination of Carriage Return and Line Feed

Chapter

3 TTL Interface

The TTL Interface provides communication between probe stations and testers which use TTL-type communications. Communication is based on a set of standard signals transmitted to the test hardware via standard Digital I/O cards (DIO). This interface is used for backwards compatibility.

Protocol

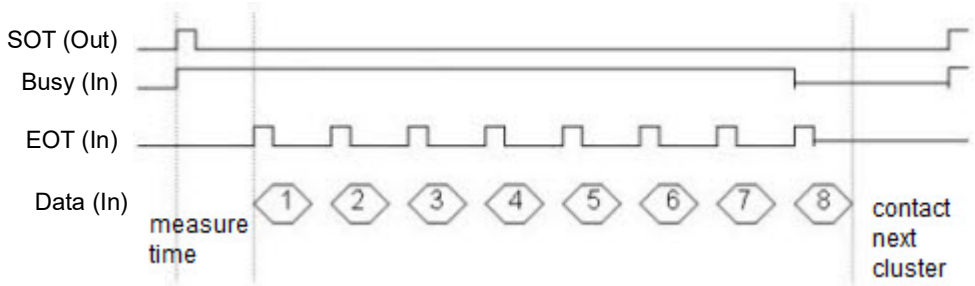
Communication is based on a simple standard test procedure that involves minimal two-way communication between the test system and the control computer. The TTL interface defines the following signals.

Standard Test Procedure

Binning Order During Cluster Measurements

SOT	Start Of Test	Sent to indicate the start of a test. The chuck has moved into the correct position and the needles are in contact.
AFD	At First Die	Sent to indicate the chuck has moved to the first die or cluster when testing complete wafers.
EOW	End Of Wafer	Sent to indicate the completion of wafer processing.
BUSY	Tester Busy Signal	Sent indicate a test is in progress. The TTL interface may not take actions other than reading results as long as the tester is busy.
EOT	End Of Test (IRQ)	Received by the TTL interface from the test software. Indicates that a single data item can be read from channel A of the DIO card. This data item contains a binning for either a single die or a die that is part of a cluster measurement. Internally this line is an interrupt line.

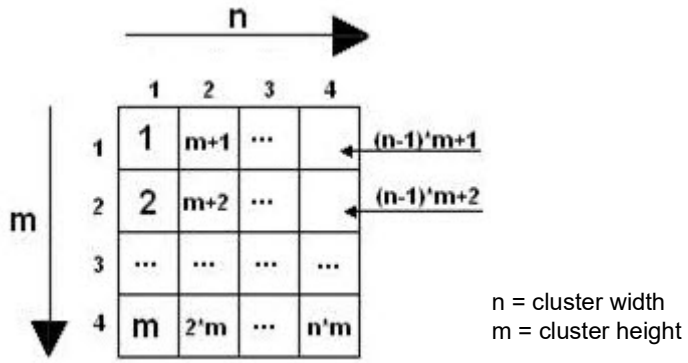
The following diagram gives an example of the two way communication going on during a test of a cluster containing 8 dies.



After the cluster has been contacted, an SOT signal is sent to the test system. The test system starts the measurement and sets the persistent BUSY signal to high. After each measurement, the test system sends an EOT signal indicating the availability of a new data item.

Binning Order During Cluster Measurements

During multiple die measurements, the test system will send one EOT for each data item in the current cluster. The order of data transmission for a single cluster is illustrated below.



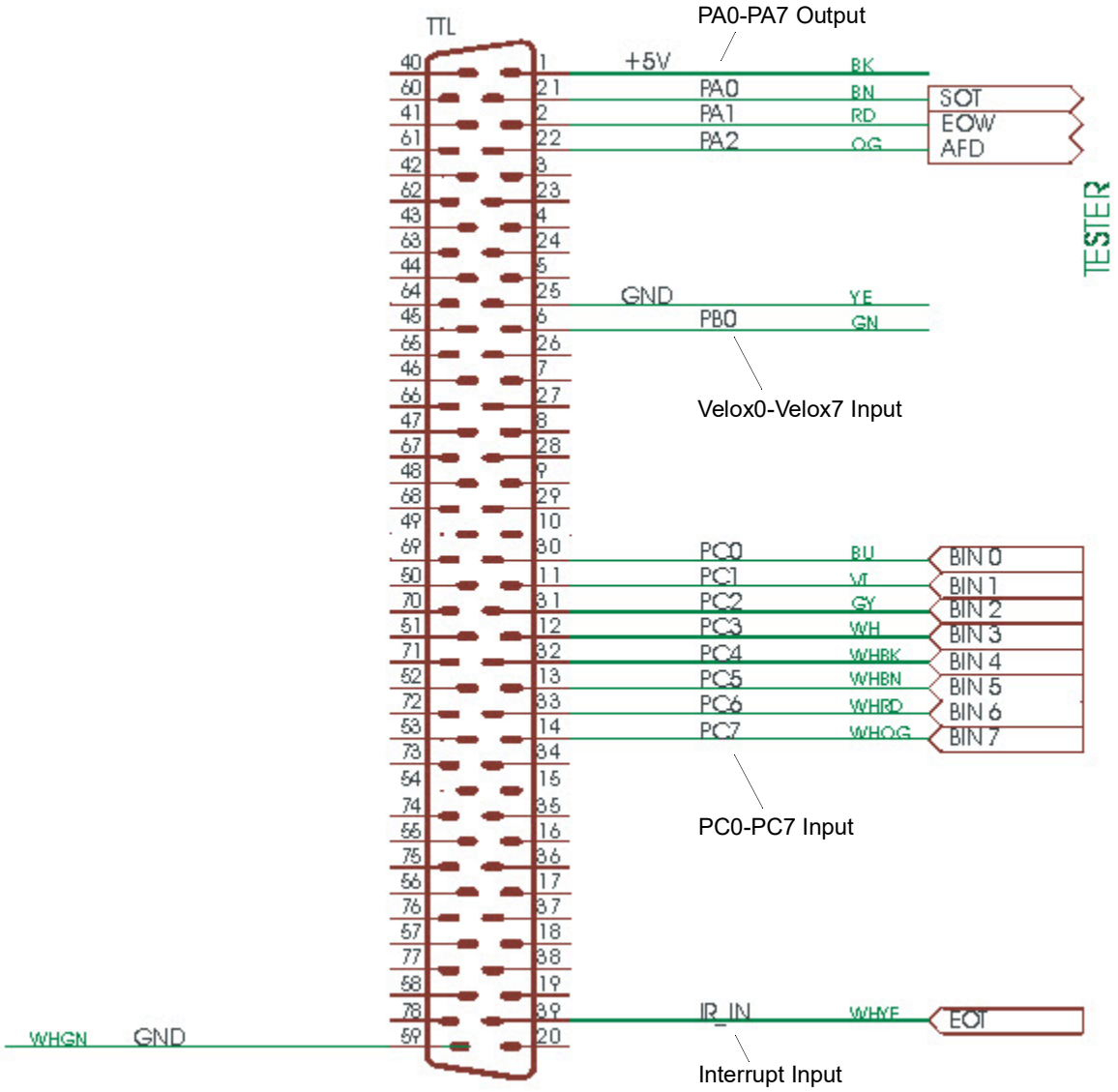
The binning consists of an 8-bit value which can be read from the DIO card (channel A). This value must be readable as long as the next test is in progress (until the next EOT). It will be overwritten by the next test result. Special care must be taken that the reading of the data does not consume more time than the testing process itself.

In such cases, the data would be overwritten prematurely causing a data underflow error in the TTL interface. After emitting the last EOT signal of a cluster measurement, the test system is setting the BUSY line to low.

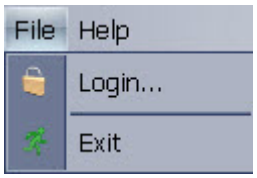
The prober is now ready to contact the next die. With the next cluster, the cycle starts again until the TTL interface emits the EOW signal indicating completion of a wafer test.

The TTL interface supports both single and cluster measurements. The type of measurement is determined by the current WaferMap cluster settings. No user interaction other than setting up a proper WaferMap is necessary to configure the TTL interface for multi-die testing.

TTL Cable Pinout



User Interface

File menu		**Login** Opens the login dialog box. **Exit** Exits the TTL Interface application

Help menu		**TTL** Opens the TTL Help file. **Contents** Displays the Contents page for a view of the topics. **Index** Displays the Index page for a complete listing of topics and keywords. **About** Displays product, version and copyright information

TTL functions are accessed using the main window and tabs. Panels are visible depending on the user access level and TTL interface configuration.

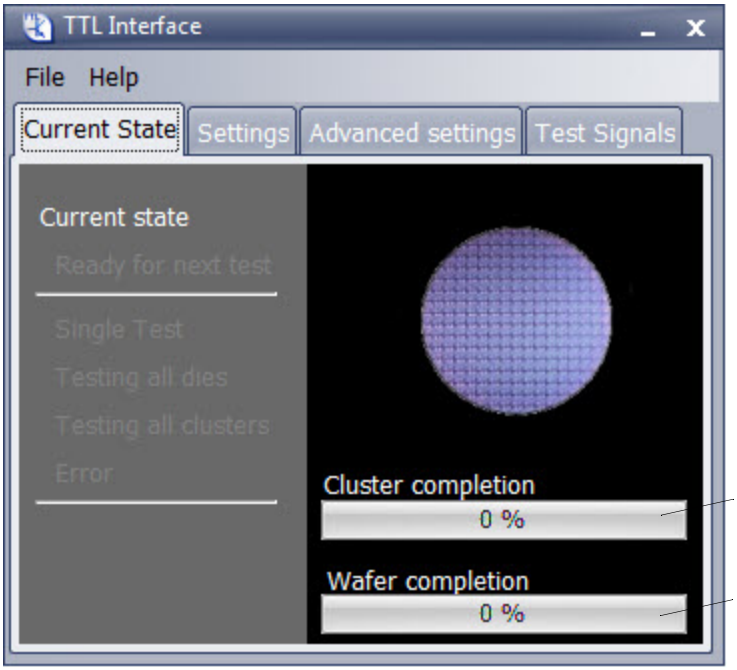
Current State	Always visible
Settings	Visible only for SuperUser and Service
Advanced Settings	Visible only for SuperUser and Service
Test Signals	AVisible only for SuperUser and Service

Current State

Current Tasks

TTL Interface Tasks

Current State provides information on the current task and general progress information.



Progress bar shows completion information if clusters are enabled

Shows overall wafer progress if testing complete wafers

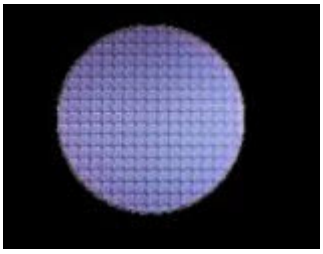
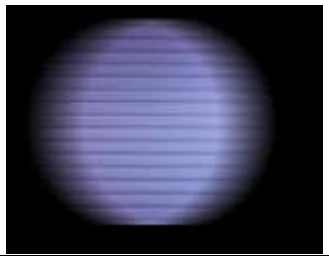
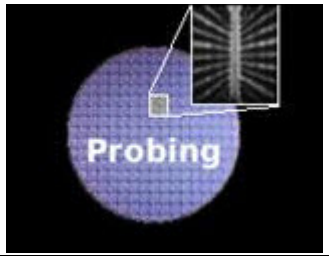
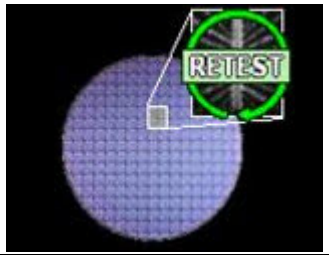
Current Tasks

Commands	Tasks	
—	Ready for next test	TTL interface is idle and waiting for its next assignment
DoTTLTest	Single test	Both single cluster or single die depending on wafer map cluster settings
StartTTLTest	Testing all dies	Clusters are disabled in wafer map

StartTTLTest	Testing all clusters	Clusters are enabled in wafer map
—	Error	One of the previous tasks caused an error and was therefore canceled

TTL Interface Tasks

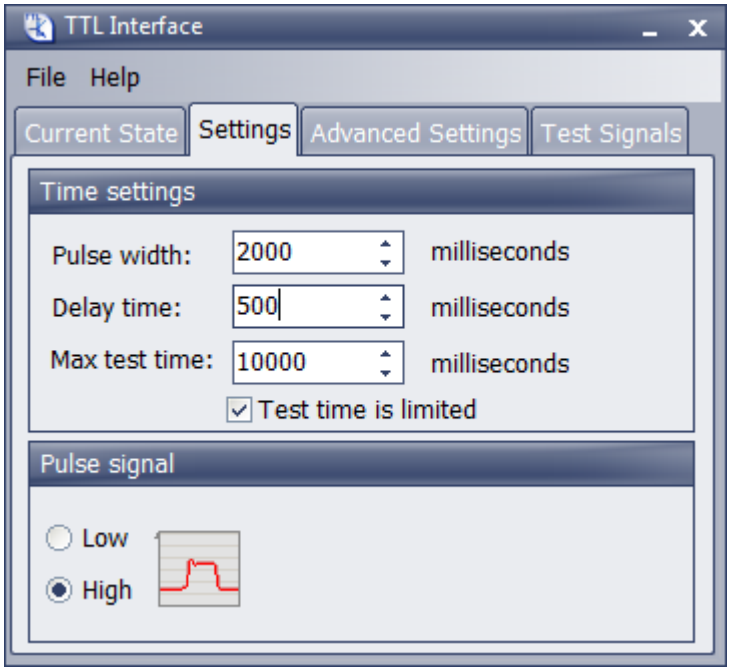
The lower part of the right window side contains two progress bars illustrating the overall wafer progress (only if testing complete wafers) and cluster completion information (only if clusters are enabled).

Idle		Interface is waiting for commands
Wafer movement		A StepNextDie or StepNextCluster command has been issued internally.
Test in progress		The TTL-Interface is waiting for EOT-Signals, ready to collect data.
Retesting cluster		Retesting cluster data due to missed EOT signals from the test system. Only possible when observing Busy-Line and Retest Option in the advanced Settings Tabsheet is selected.
Test Done		Displayed after finishing a wafer test or a single test.

Error		A previous command caused an error that lead to canceling the process

Settings

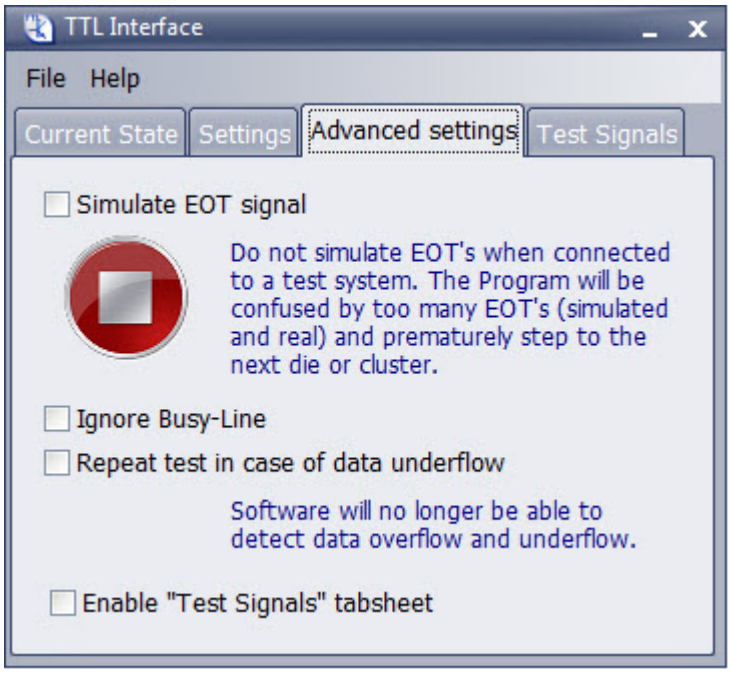
The program automatically loads the selected options at startup, and saves them before exiting. The file <Velox folder>\ configuration.txt is used for this purpose. See TTL Interface Commands.



Pulse Width	Time duration the program output signal persists before being cleared. Most testers are able to read a signal smaller than 10 ms. High precision of the pulse width is guaranteed for values ≤ 2ms. A very high CPU usage can extend the width, but the pulse will not be as small as the entered value.
Delay Time	Time duration the program waits after a test complete signal is received before sending a new test start signal (SOT).
Max test time	Time duration the program waits for the EOT signal before canceling the test.
Pulse signal	Use this option to change the logical level of the pulse. Low – Logical 0 is Low High – Logical 1 is High
EOW signal type	Option allows changing the type of EOW signal. Pulse – Short signal (with width equal to “Pulse width”) Persistent – Long (infinite) EOW signal.

Advanced Settings

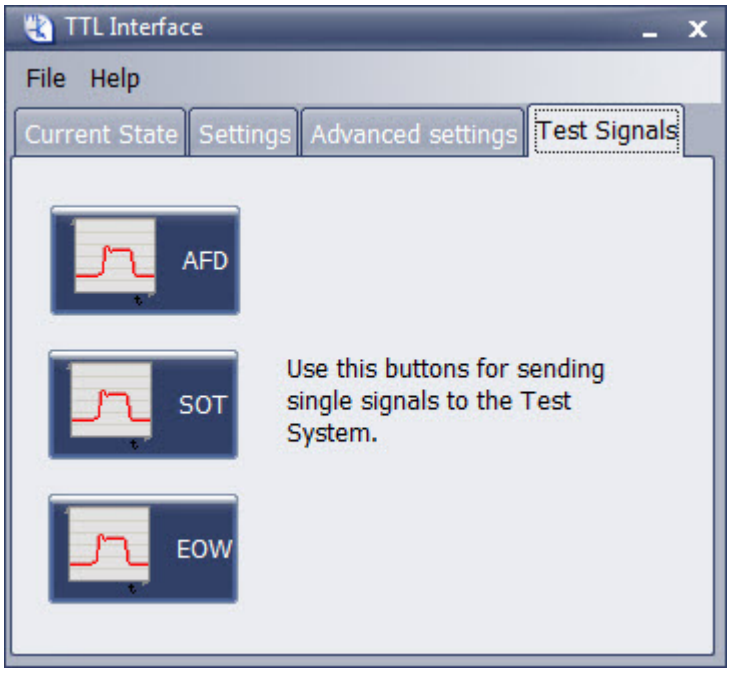
This view is available exclusively to users with high access levels. It allows a SuperUser to activate a demo mode and to enable or disable TTL tabsheets.



Simulate EOT signal
When simulating a test system using TTL Terminal Box, this switch allows an experienced user to simulate incoming EOT-Signals. The signal emission will be done by a timer with an interval of 50 ms. EOTs do not have to be raised manually which would permit test processing of complete wafers.
Ignore Busy-Line
Affects only the measuring of cluster data. Enabling this switch stops the tester from querying the Busy-Line. Internally querying the Busy-Line is not necessary. The TTL interface counts all incoming EOT signals and compares the number of received data with the numbers expected according to the current WaferMap layout and cluster data. If this option is enabled, the TTL interface proceeds to the next cluster or die after having received all expected data items, regardless of the BUSY-Line state. If it does not receive the expected number of data it either waits indefinitely or until the predefined timeout occurs. If the tester is sending more data than expected, the TTL interface steps WaferMap after having received the number of data items that match the number of dies per cluster, whether or not the tester signals completion. The Busy-Line helps in finding problems if the test system sends fewer or more data than expected. Querying the Busy-Line activates error messages that indicate data underflow or data overflow.
Repeat test in case of data underflow
Affects only the measuring of cluster data. It is available only when observing the busy line. If enabled, the tester will submit a new SOT signal to the test system instead of emitting an error in case of a data underflow condition. Data underflow conditions occur if missing EOT signals are sent by the test system. This can happen during periods of increased system load, such as when running time-consuming applications in addition to the TTL interface.

Test Signals

Test Signals can be used to manually send standard signals to the test hardware. This may be useful for debugging applications running on the test system.


TTL Interface
File
Help
Current State
Settings
Advanced settings
Test Signals
AFD
SOT
EOW
Use this buttons for sending single signals to the Test System.

Chapter

4 GPIB Interface

The GPIB Interface is the GPIB connectivity application. It requires a National Instruments PCI-GPIB board and the associated software drivers for the Windows platform (GPIB-USB-HS or PCI-GPIB Board and NI-488.2M software). The Velox application incorporates its own GPIB hardware driver. The client PC, which is acting as the system controller, must have the GPIB card configured as the Controller-In-Charge (CIC) of the bus.

Overview

The GPIB interface allows a client computer (configured as the GPIB controller) to access and manipulate both the Velox software and the prober firmware using the Velox Remote Command set (in ASCII-text format). It can be used for writing a custom, GPIB-enabled program which will coordinate and control a test instrument and prober over a GPIB bus.

The GPIB interface appears to the controller as a device on the GPIB bus, addressable through its assigned Primary Device Address (PAD). It routes a remote command from the GPIB bus to the Velox server; then it is dispatched either to a Velox application (as with the WaferMap command “StepNextDie”) or directly to the prober (as with Kernel command MoveChuck). The command response is then routed in the reverse direction from the appropriate command target back to the GPIB bus.

To enable GPIB interface, open the Velox Setup application, select the PC Config tab and check NI GPIB in the Options section. The interface may also be configured using the Velox GPIB Interface application.

Message Server Command Processing

When the client’s GPIB board sends a text-string command across the bus, it is received by the Velox GPIB interface software, which forwards it to the Message Server. The Message Server then forwards the command to the controller hardware or other Velox software applications, as appropriate for the particular command.

The ultimate target (software or hardware) of the command attempts to execute it, then issues a text-string response. This response traverses the same path in reverse: the target sends it to the Message Server, which then sends it to the Velox GPIB interface software, which (optionally) sends it across the GPIB bus to the client computer.

Protocol

Termination
Command Protocol for SCI type commands
Serial Polling

Termination

Many GPIB communication problems are caused by inconsistent termination settings between the PC or instrument (configured as the GPIB Controller) and Velox (typically configured as a GPIB Device).

The termination settings for the Velox GPIB Device are set in the configuration window of GPIBInterface.exe. You can select None or one of the predefined response terminators.

ASCII Decimal	Control character
13	CR – Carriage return
10	LF – Line feed
	CR/LF – Combination of Carriage Return and Line Feed

The selected response terminator will be added to the response string. GPIBInterface.exe always works in End of Transmission (EOT) mode, which is referred to as “Send EOI at end of Write” by NI. The GPIB EOI line will be asserted at the completion of every write operation. Whether or not a termination character is used, the End of Transmission (EOT) mode is recommended.

Command Protocol for SCI type commands

The GPIB interface follows the Talker/Listener GPIB device paradigm by monitoring the Talk and Listen bus lines. When the controller addresses its PAD and turns on the Listen line, GPIB will read a string from the bus, and interpret it as a Velox Remote Command string. When the Listen line toggles off and the Talk line toggles on, GPIB will write the Command Response string back to the bus as soon as the response is available. Every remote command returns a Response string, starting with an error code, even if it does not contain any additional response parameters.

After sending a Command string to the GPIB interface, the GPIB controller must request the Response string from GPIB before sending new commands. The implemented output queue can contain up to 100 responses before it starts deleting the oldest.

All commands and responses are ASCII text strings. All commands and responses, and the respective data arguments, are separated by spaces. Commands are not case sensitive.

Commands can either be sent synchronously (Normal) or as a post command.

Normal (synchronous)	The Response string is returned after the command is completed.
Post command	The response returns immediately, but consists of the string PostCommand Sent and does not have any further information.

If a command is sent as post command, the command string must be prefixed with the ‘<’ character.

A command string consists of a command name followed by a space, then followed by any data necessary, separated by spaces. The exact data depends on the command. See the remote command set definition for a list of commands that may be sent. Data is either decimal (float) or whole (char, Unit, Int) numbers. The command ID number can be sent instead of the command name.

Command String

Description

Moves the chuck to the specified position.

Example

```
34 1233.5 -67745.3
0:
```

If no error occurs during the command execution a normal response string is generated from the prober, if requested. The response normally consists of a zero followed by an error separator Remote_Error_Separator character, usually a colon. If the command executed returned data (for example, ReadChuckPosition), this data will appear after the error separator.

Normal Response String

Description
Returns the actual chuck position in x, y and z.
Example
`ReadChuckPosition` `0:123.4 -567.8 1022.8`

In case of an error situation the response string will differ from that specified in the remote command set.

The response consists in this case of an error# (a whole number between 1 and 999, see the Error Codes for more details) followed by an error separator Remote_Error_Separator, followed by the error source and the error text.

Error Response String

Description
Error# 75 on an expected software fence violation after the chuck movement.
Example
`MoveChuckIndex 1 0 R Y 100` `75:[Chuck] Software fence error`

The post command response returns immediately (before the command is done) and does not contain further information.

Post Command Response String

Description
Returns post command response.
Example
`MoveChuckIndex 1 0 R Y 100` `0:PostCommand sent`

The error separator is the character that appears between the 0 (no error) and the returned data in a command response or between the error number and the error text and the original command in an error response. It can be either a space or a colon (default is colon).

Error Separator

Description
Response with Error Separator space.
Example
`ReadChuckPosition` `0 0.5 1.8 14.7`
Description
Response with Error Separator colon.

Example
`StepProbeSite 1 13` `82:X Axis on End Limit:StepProbeSite 1 13`

Serial Polling

NOTE

This section is not applicable if TSK emulation is ON. In that case, the TSK protocol for status byte is followed instead. See Board Configuration for more information on TSK emulation.

When the Legacy SCPI Status Byte is set to ON, the meanings of the three low order bits are changed. The Legacy SCPI Status Byte should be set to OFF unless the controlling program specifically uses those legacy low order bits from Nucleus.

The status word, read by serial polling, consists of the following bits:

Bit 0	RDY – GPIB is ready	0 – A command is in process or data transmission is in progress. 1 – GPIB interface is ready to receive a new command. Before sending a new command, check that a response is still available.
Bit 1-3	not used	
Bit 4	RSP – Command response or data	0 – No response or data is available. 1 – A command has been processed and a response or data is available. The bit is reset when a new command is sent or the response or data has been read.
Bit 5	ESB – Event status byte	0 – All bits of the masked event status register are 0. 1 – At least one bit of the masked event status register is 1. The bit is dependent on the event status register status (GPIB commands *ESE, *ESE?, *ESR?).
Bit 6	SRQ – Service request	0 – Service request is not set. 1 – Service request is set. The bus controller may request the SRQ be set after command execution. If the SRQ is set, it can be defined in the GPIB configuration window.
Bit 7	not used	

GPIB Commands

NOTE

If TSK emulation mode is enabled (ON), then none of these commands will work. See Board Configuration for more information on TSK emulation.

Command	Command Type	Implemented
*ADD	Command; auto configure	No
*CAL?	Query; Internal Operation	No
*CLS	Command; Status & Event	Yes

*DDT	Command; Trigger	No
*DDT?	Command; Trigger	No
*DMC	Command; Macro	No
*EMC	Command; Macro	No
*EMC?	Query; Macro	No
*ESE	Command; Status & Event	Yes
*ESE?	Query; Status & Event	Yes
*ESR?	Query; Status & Event	Yes
*GMC?	Command; Macro	No
*IDN?	Query; System Data	Yes
*IST?	Query; Parallel Poll	No
*LMC?	Query; Macro	No
*LRN?	Query; Internal Operation	No
*OPC	Command; Synchronization	No
*OPC?	Query; Synchronization	No
*OPT?	Query; System Data	No
*PCB	Command; Controller	No
*PMC	Command; Macro	No
*PRE	Command; Parallel Poll	No
*PSC	Command; Status & Event	No
*PSC?	Query; Status & Event	No
*PUD	Command; System Data	No
*PUD?	Query; System Data	No
*RCL	Command; Stored Settings	No
*RDT	Command; System Data	No
*RDT?	Query; System Data	No
*RMC	Command; Macro	No
*RST	Command; Internal Operation	Yes
*SAV	Command; Stored Settings	No
*SDS	Command; Stored Settings	No
*SRE	Command; Status & Event	Yes
*SRE?	Query; Status & Event	Yes
*STB?	Query; Status & Event	Yes
*TRG	Command; Trigger	No
*TST?	Command; Internal Operation	Yes
*WAI	Command; Synchronization	No

*CLS Command

Description

The Clear Status (*CLS) command clears the bits of the Standard Event Status Register, the output queue, and the Status Byte Register, but sets the RDY-bit to true.

Example

```
*CLS
```

*ESE Command

Description
The Standard Event Status Enable (*ESE) command sets the Standard Event Status Enable Register, which is used for masking the Standard Event Status Register. Enabled Standard Event fields can lead to setting the ESB bit of the Status Byte.

Parameter Input
Standard Event Enable Bits

Example
```
*ESE 8
```

*ESE? Command

Description
The Standard Event Status Enable Query (*ESE?) command reads the content of the Standard Event Status Enable Register.

Parameter Output
Standard Event Status Enable Register

Example
```
*ESE?
8
```

*ESR? Command

Description
The Standard Event Status Register Query (*ESR?) command reads the current content of the Standard Event Status Register. The register is cleared by reading.

Parameter Output
Standard Event Status Register

Example
```
*ESR?
0
```

*IDN? Command

Description
The Identification Query (*IDN?) command reads identification values from the device. The response has the following form:
<Manufacturer>,<Model>,<Serial number>,<Firmware level or equivalent>

Parameter Output
Identification

Example
```
*IDN?
"Cascade Microtech,Velox,0,1.0.2.31"
```

*OPC Command

Description
*OPC sets the "Operation Complete" bit in the Event Status Register. If in Legacy Status byte mode, then the Done (DNE) bit in the status register is also set.

Example

```
*CLS;:mov:next:die;*OPC
Read Event Status Register with *ESR? until the OPC bit is set in order to wait for
the move to complete.
```

*OPC? Command

Description

Places a 1 in the output queue.

Parameter Output

1

Example

```
*CLS;:move:next:die;*OPC?
1
```

```
*CLS; <clear all>
:move:… (any move command)
```

Note that Legacy GPIB string response must be OFF.

```
*OPC?
Wait for the "1" response from the *OPC? query to be sure the move is finished
before sending more commands.
```

*RST Command

Description

The Reset (*RST) command sets the following registers:

- Standard Event Status Register (ESR) will set the 'Operation Complete' bit
- Standard Event Status Register Enable (ESE) will be zero
- Service Request Enable (SRE) will be zero
- Status byte (Serial Poll) will set bit 0 (Ready)
- The station then goes into listener mode.

Example

```
*RST
```

*SRE Command

Description

The Service Request Enable (*SRE) command sets the Service Request Enable Register, which is used for masking the Status Byte for Service Request. Enabled bits of the Status Byte can lead to a Service Request.

Parameter Input

Service Request Enable Bits

Example

```
*SRE 16
```

*SRE? Command

Description

The Service Request Enable Query (*SRE?) command reads the content of the Service Request Enable Register.

Parameter Output

Service Request Enable Register

Example
SRE?
48

*STB? Command

Description
The Read Status Byte Query (*STB?) command reads the content of the Status Byte Register.

Parameter Output
Status Byte Register

Example
```
*STB?
48
```

*TST? Command

Description
This Self Test Query (*TST?) command causes an internal self test of the device.

Parameter Output
Error code
0 if no error occurred

Example
```
TST?
0
```

Asynchronous GPIB communications

Asynchronous GPIB communication can be used to avoid the GPIB interface timeouts for SCI commands that can take very long (> 10 minutes).

$StartCommand

Description
Executes the provided string on a separate thread and returns immediately.

Parameter Input
A single string parameter. Every character after the initial space is sent as a command.

Parameter Output
If the operation starts successfully then the string COMPLETE is returned. Otherwise and error string is returned.

Example
```
$StartCommand StepNextDie
$StartCommand StepNextDie 1 3
$StartCommand MoveChuck 7500 5400 R Y 100
$StartCommand :move:absolute 2 10000 10000 none
```

$CommandStatus

Description

Returns either $BUSY or the string response of the command that was issued with the $StartCommand operation. $BUSY is returned if the command is still executing.

Example

```
$StartCommand StepNextDie
COMPLETE
$CommandStatus
$BUSY
$CommandStatus
0:
```

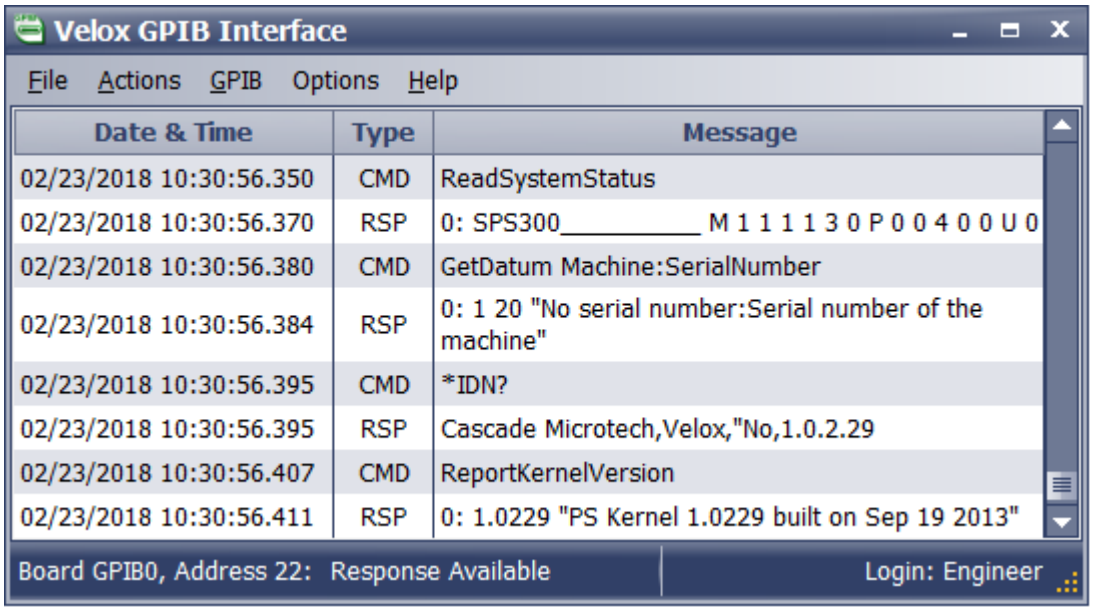
Example Sequences

Example 1	`$StartCommand StepNextDie`
	`COMPLETE`
	`$CommandStatus`
	`$BUSY`
	`$StartCommand MoveChuck 7500 5400 R Y 100`
	`Error: GPIB is busy processing the command: StepNextDie`
	`$CommandStatus`
	`0:`

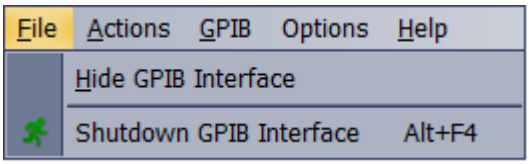
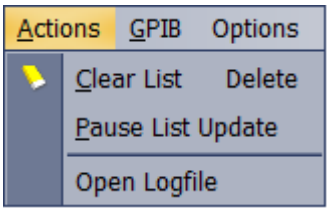
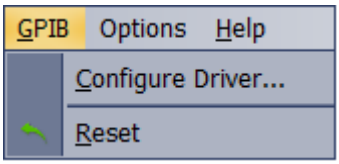
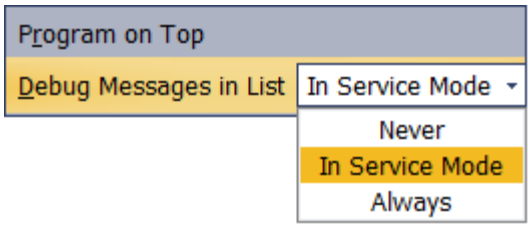
Example 2	`$StartCommand :move:abs 2 10000 10000 none`
	`COMPLETE`
	`$CommandStatus`
	`$BUSY`
	`$CommandStatus`
	`$BUSY`
	`$CommandStatus`
	`COMPLETE`

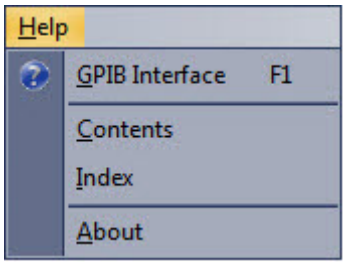
User Interface

The Velox GPIB user interface displays the command and response strings routed across the GPIB bus, and allows configuration of the device. This window does not have to be visible for the interface to be active.



The menu bar contains the functions of the Velox GPIB interface application and many system-level control and setup functions.

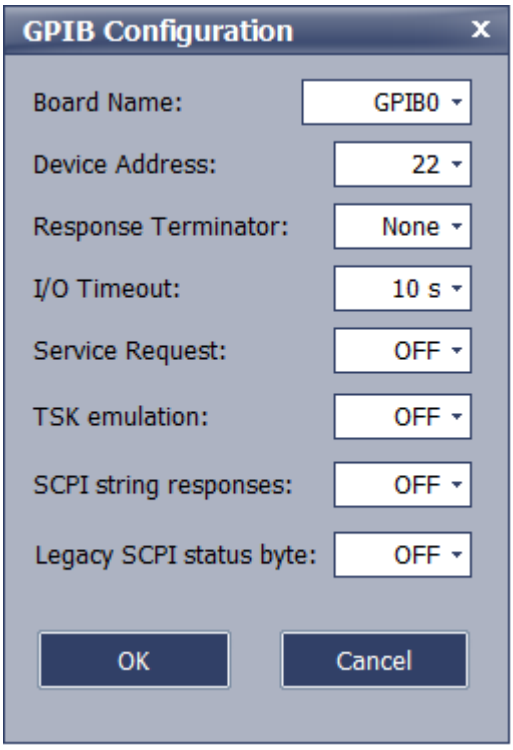
	**File menu**
	Hides the GPIB interface.
	Terminates the communication.
	**Actions menu**
	Clears the log window.
	Select Pause List Update to temporarily pause the scrolling list of GPIB activities. It makes it possible to stop and read something when trouble shooting.
	Opens the log file where the list of GPIB activities has been written.
	**GPIB menu**
	Configure Driver opens the GPIB configuration window.
	Reset reinitializes the GPIB communication.
	**Options menu**
	Select Program On Top to have the GPIB interface window always be on top of any other window on the screen
	Select when Debug Messages are added to the scrolling list of activity. • Never add debug messages to the list. • In Service Mode (applies only to Service access) • Always add debug messages to the list.



Help menu
Opens the Help file to the GPIB page.
Displays the Contents page for a view of the topics.
Displays the Index page for a complete listing of topics and key words.
Displays product, version and copyright information.

Board Configuration

GPIB configuration sets up the GPIB (IEEE-488) interface.



Board Name	If the PC is equipped with more than one GPIB card, Board Name specifies the card used by GPIBInterface.exe. The first one is always called GPIB0, the second GPIB1, and so on
Device Address	Specifies the bus device address of the GPIB card used by GPIBInterface.exe. The client computer uses this device address to communicate with Velox. • The Velox default device address is 22. • To match the Nucleus default, change address to 28.
Response Terminator	Specifies the string to append to a response (typically, None).
I/O Timeout	This is the timeout value sent to the GPIB driver. It determines how long Velox will wait for a controller to read a response termination character.
Service Request (SRQ)	ON—A SRQ is generated every time a response is available or the RDY bit is set in the status byte OFF—Standard 488.2 SRQ behavior is used, which means that the *SRE and *ESE commands need to be used to define when a SRQ is generated.

TSK emulation	Turn TSK emulation mode ON to make Velox expect and require TSK style commands and set the status byte accordingly. In TSK mode, no other SRQ settings apply and no response strings are sent, other than to queries. Note that the 488.2 commands (for example *IDN?) do not work in TSK mode.
SCPI string responses *Applies to legacy SCPI only. See Legacy Commands: SCPI for more information.*	The string-return method of handshaking is the simplest form of remote communication. When using this form of handshaking, you send a command to the specified instrument and wait for a response. After you receive an appropriate response, you send the next command. The response strings to commands must be enabled with the following meta command: `$:set:resp on` The `$:set:resp` command instructs the probe station to return an ASCII string after each command is successfully completed (or an error occurs). After each command is successfully executed, the probe station returns the ASCII string “COMPLETE”. If an error occurs, the probe station returns the ASCII error message instead. Also, if the command was a query, the probe station returns the response only (or the error message only). The first character in an error message is always the @ (“at”) sign. For GPIB communication, the string-return method of handshaking relies on a timeout value. The timeout value on the host indicates to the host how long the host should wait for a response from the probe station.
Legacy SCPI status byte *Applies to legacy SCPI only. See Legacy Commands: SCPI for more information.*	When it is set to ON, the lowest three bits in the status byte are set with legacy conventions and when set to OFF, these three bits are set with Velox conventions. Only the application specific bits 0-2 are affected. The table below show the bits that are different when the Legacy SCPI status byte is set to ON. Bits 3-7 are the same as the standard Velox status byte. See Serial Polling.

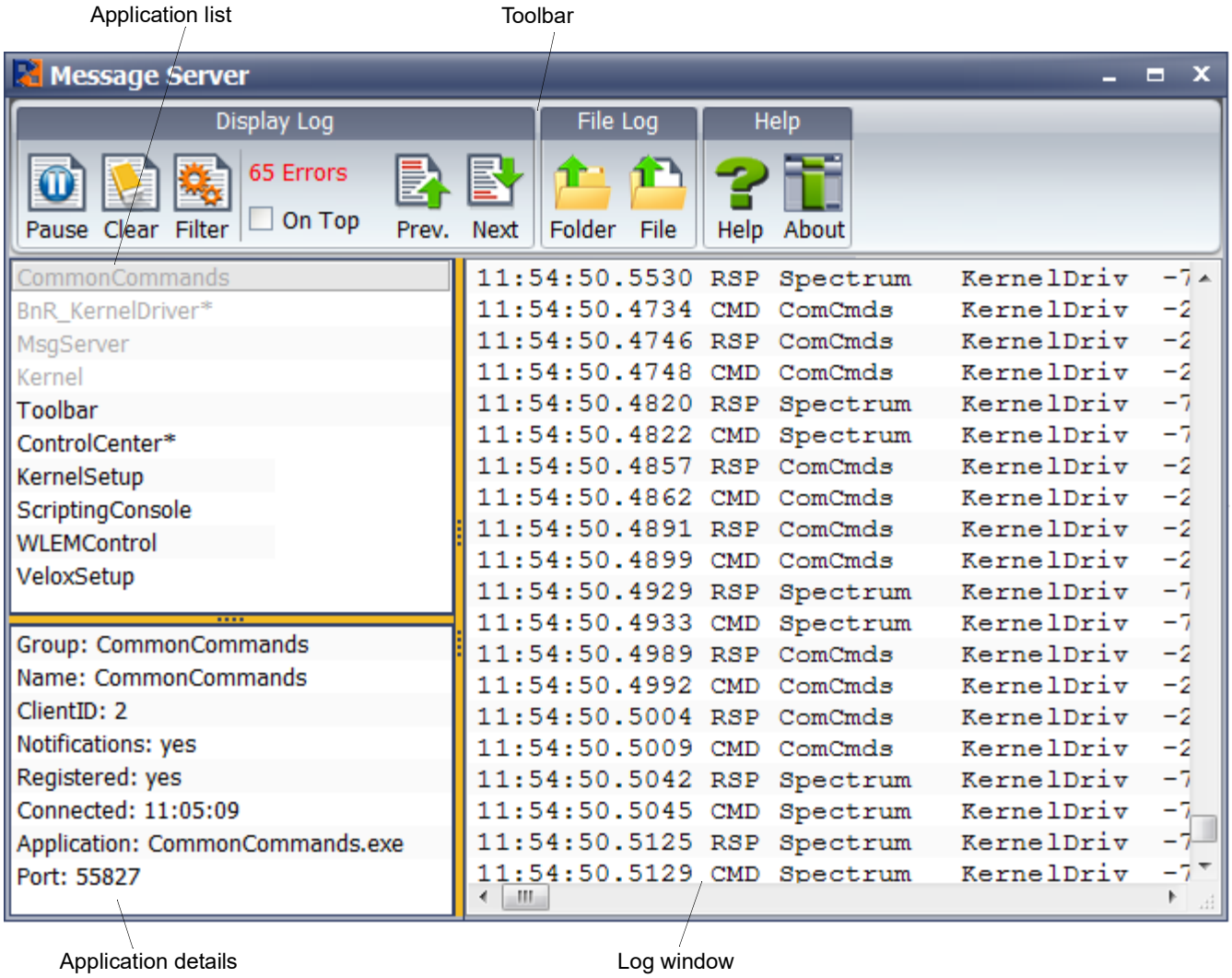
Bit 0	ERR – Error If this bit is ON, an error has occurred.
Bit 1	DNE – Done bit This bit gets set when an operation is complete (OPC).
Bit 2	ONL – Online bit When this bit is ON, the station is ready to receive commands.

Chapter

5 Message Server

The Velox product suite consists of multiple applications communicating with each other. The Message Server is the central application responsible for dispatching the messages between applications.

The Message Server window enables Message Server configuration, lists all connections, and displays all event logs.



For more information, see:

- User Interface
- Connecting to the Message Server

User Interface

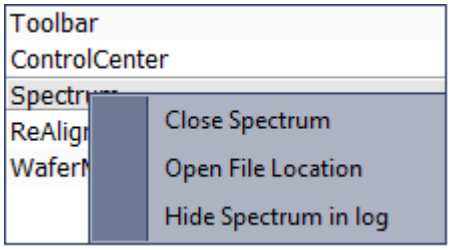
Toolbar
Applications
Message Log
Log Filter Configuration

Toolbar

Pause/Continue		• Click Pause to temporarily stop logging. Adds no new logs to log window. Logging continues in background. • Click Continue to continue adding new logs.
Clear		Clear the current log.
Config		Configure the log filter options.
Previous Error		Select previous error.
Next Error		Select next error.
# Errors		Displays the number of errors in current log list.
On Top		Toggle Message Server display.
Open File		Display the saved log file.
Open Folder		Open the MessageServer log folder.
Show Help		Display the Help file.
About		Display the About panel.

Applications

The applications list and details panels are located below the toolbar. Left-click to show details about an application in the details panel. Right-click to open a context menu for the application.



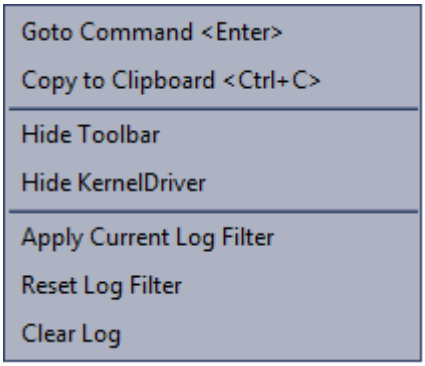
Close Close this application by an SCI command.	*Admin only.*
Start Start this application as a new process.	*Available only if a mandatory application is missing.*
Open File Location Open the folder containing this application.	*Available only for local processes using the correct registration process.*
Hide in Log Messages from and to the selected application will not show up in the log window.	

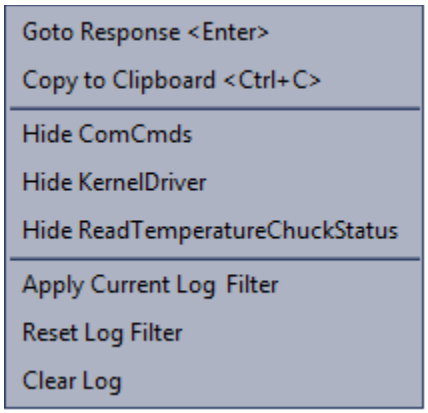
Group: Spectrum Name: Spectrum ClientID: 7 Notifications: yes Registered: yes Connected: 14:29:42 Application: Spectrum.exe Port: 55971	This panel shows details about the currently-selected application. • Displays the the Name and Group in the SCI protocol • Displays the internal number (ClientID) to enumerate connections • Verifies whether notifications are enabled for this connection • Verifies whether this application is connected correctly • Displays the time the application connected • Displays the application's executable file • Displays the port number of the connected socket

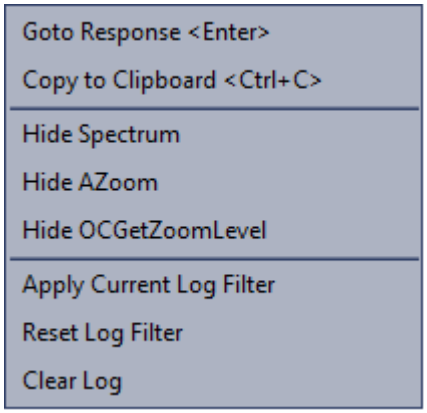
Message Log

Left-click a command to highlight its response in the list. Right-click to open a context menu. The context menu selections depend on what applications are open.

Goto Response	Jump to the response of this command (for commands only).
Goto Command	Jump to the command of this response (for responses only).
Hide <Source>	Hide all further log messages from <Source>.
Hide <Destination>	Hide all further log messages from <Destination>.
Apply Current Log Filter	Use the current log filter on all currently shown log-entries.
Reset Log Filter	Reset log filter to default.
Clear Log	Clear the current log window.



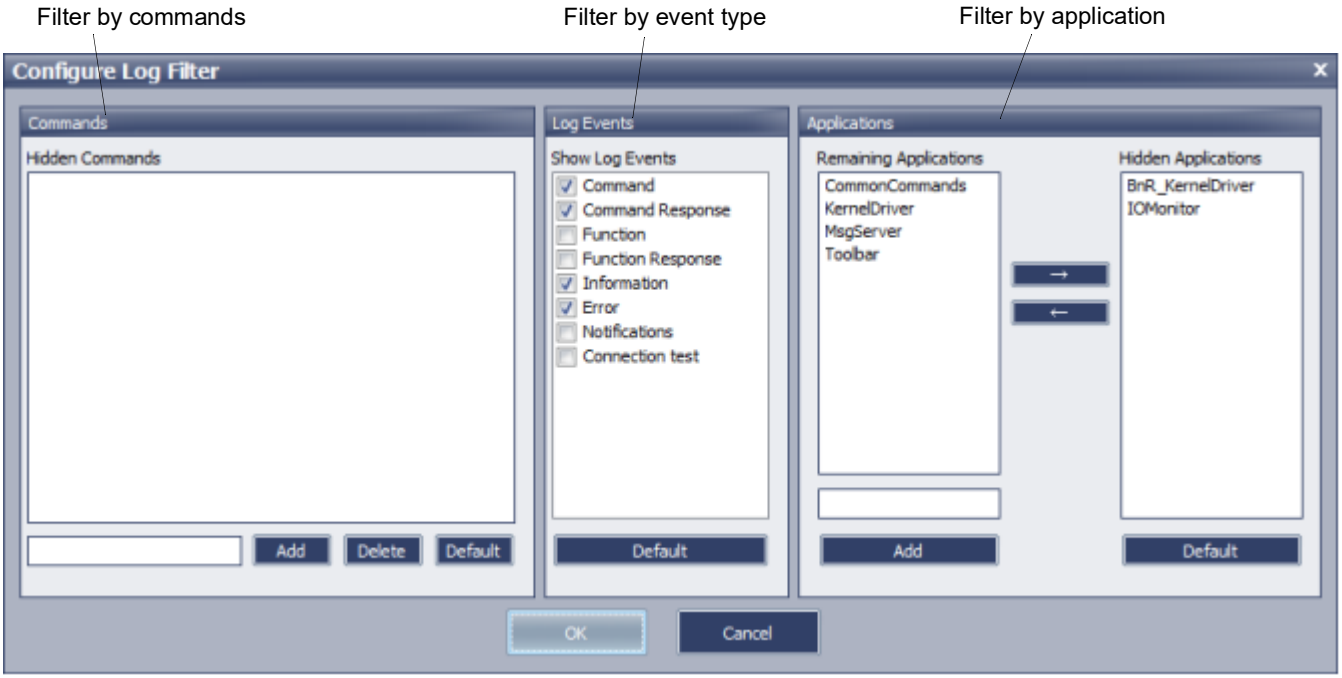




Log Filter Configuration

Logs can be filtered by command, event type, and applications.

- Click OK to save and apply the current configuration. This is saved between restarts.
- Click Cancel to return to previous settings.



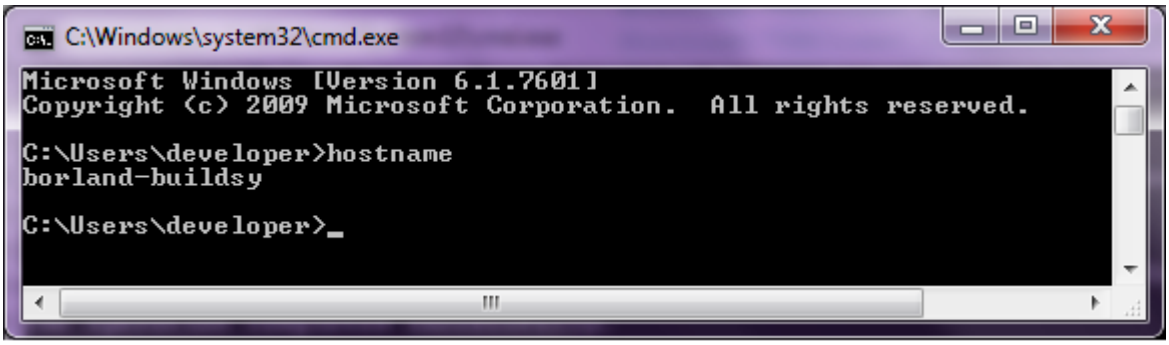
Commands Commands in this list will not be displayed in the log window.	Add	Add command typed in left of this button into the block list.
	Delete	Delete currently-selected command from this list.
	Default	Reset this list to default (empty list).
Event type Unselected (unchecked) Event types will not be displayed in the log window.	Default	Reset this list to the default value.
Application Selected applications under Hidden Applications will not show up in the log window.	Add	Add application typed in into the text field under Remaining Applications.
	Default	Reset this list to its default.
	→	Move the application selected in Remaining Applications to Hidden Applications.
	←	Move the application selected in Hidden Applications to Remaining Applications

Connecting to the Message Server

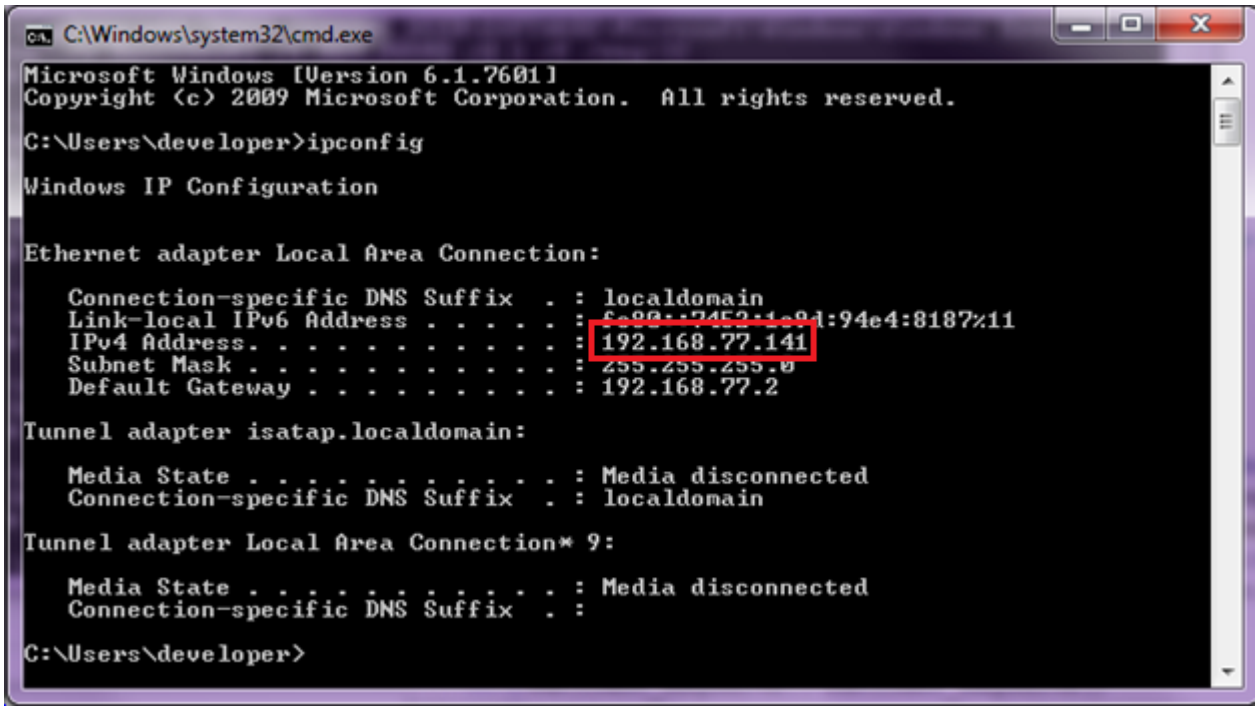
To connect to the Message Server by sockets, the hostname or IP-Address of the PC running Velox is required.

- If Velox and the controlling software runs on the same PC, the IP Address 127.0.0.1 can be used.
- If the IP Address of the PC is configured by DHCP (e.g., if it is in a domain), the hostname should be used.

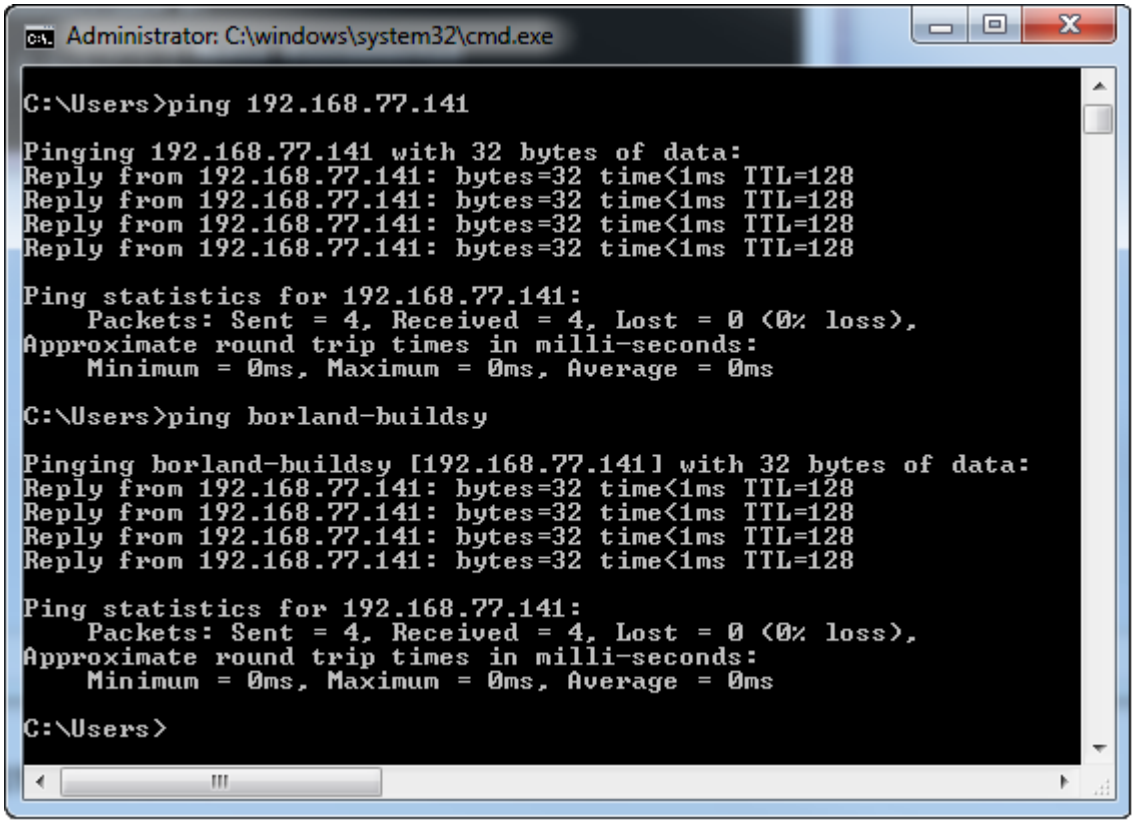
1. Open the command prompt on the Velox PC and enter <hostname>.



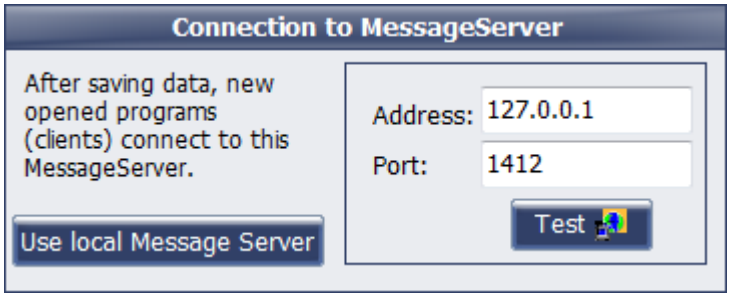
2. If the IP Address is configured statically, the IP Address can be queried from command prompt by <ipconfig>.



3. If the two PCs are connected properly, the connection can be tested by a <ping> from the controlling PC to the PC running Velox.



The configured MessageServer Port is also required. The port is 1412 by default. This can be checked with the Velox Setup tool. From the main Velox menu, click All App



With hostname and port, a connection via sockets can be made to the PC running Velox. Refer to the documentation of your programing environment or other resources for more information on how to establish and use such a connection.

NOTE

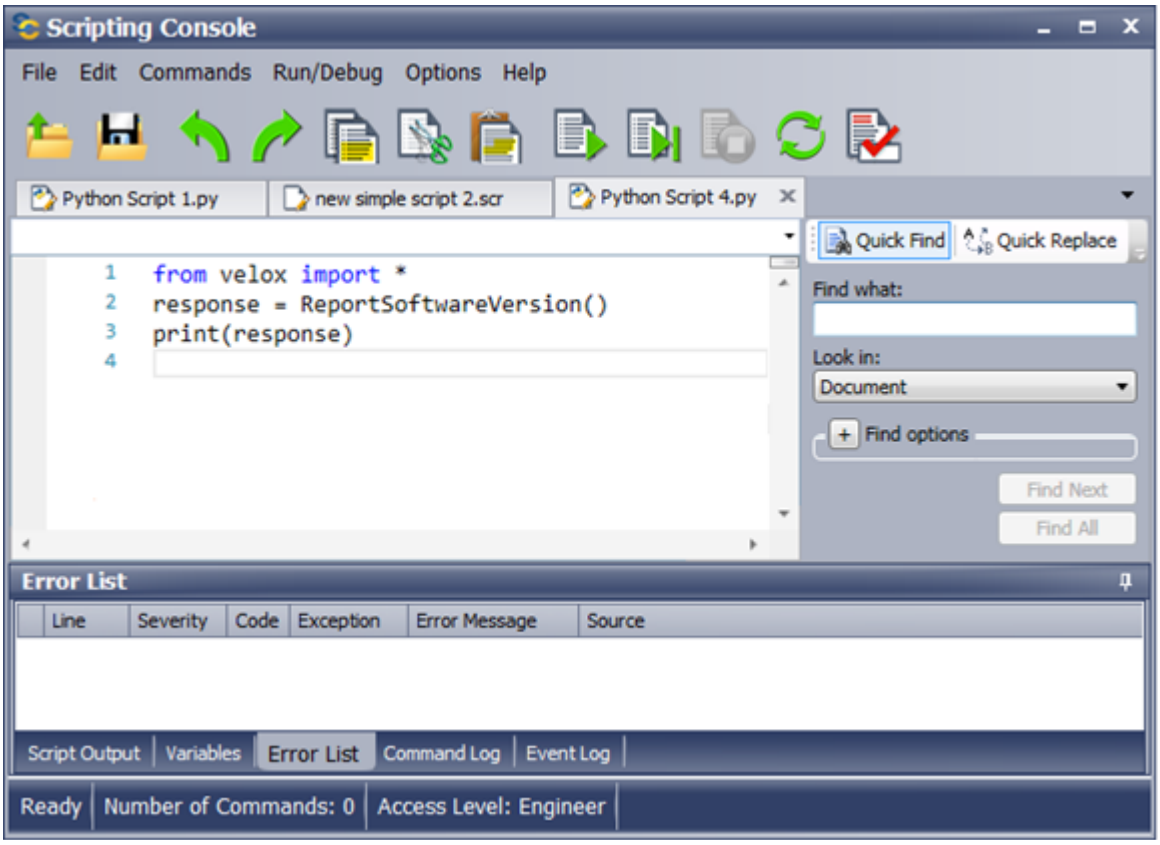
Only FormFactor supplied controllers are supported. Adding third-party software or drivers, or modifying the software installed by FormFactor may void your warranty.

The Windows firewall often hinders a proper connection. The connection can be verified by temporarily disabling it. If the firewall hinders the connection, a proper firewall rule must be configured. Refer to Windows help or other resources for instructions.

Chapter

6 Scripting Console

The Velox probe station software allows you to create task automation through a simple Application Program Interface (API) called System Control Interface (SCI).



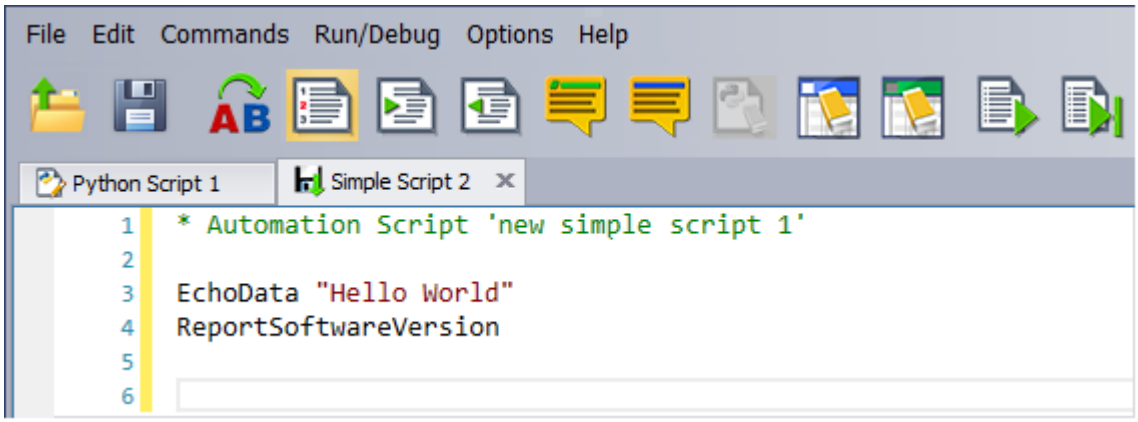
A simple example is getting the current Velox software version from the probe station by typing ReportSoftwareVersion and clicking the Run Script button. All the necessary service registration and communication protocols are handled by the Scripting Console. A simple "Hello World" example looks like this:

```
* Automation Script 'new simple script 1'
EchoData "Hello World"
ReportSoftwareVersion
```

The response from the commands is displayed in an output log within the Scripting Console application.

Scripting Console also supports fully programmable automation using the full Python programming language for complex tasks. Both the simple sequential scripting and the Python scripting languages are easy to learn and supported by the tools in Scripting Console. The Scripting Console is part of the Velox software installation.

Start Scripting Console from the Velox toolbar, or, from the Windows Start menu, select:
All Programs > Cascade Microtech > Velox > Tools > Scripting Console



The Velox Scripting Console provides an integrated development environment for editing, executing, and debugging automation scripts. Capabilities include:

- Syntax highlighting
- IntelliSense-style code completion (Python)
- Breakpoints and Single Step debugging
- A tree-structured command chooser that lists all available SCI commands
- Import and translation of legacy Remote Communicator BASIC and Pascal scripts to Python
- Conversion of simple sequential scripts to Python scripts

The SCI commands expose the APIs available in the Velox software. Over 650 different commands are available. These allow a wide range of functionality including:

- Microscope control
- Image Capture
- Platen Movements
- Manipulating Project Files
- Chuck Alignment and Movement
- Probe Movement and Measurement
- Heater Operations and Measurement
- Application Automation of Spectrum, Wafer Map, Velox Pro, and other tools

All of these commands can be combined using the scripting languages for custom applications.

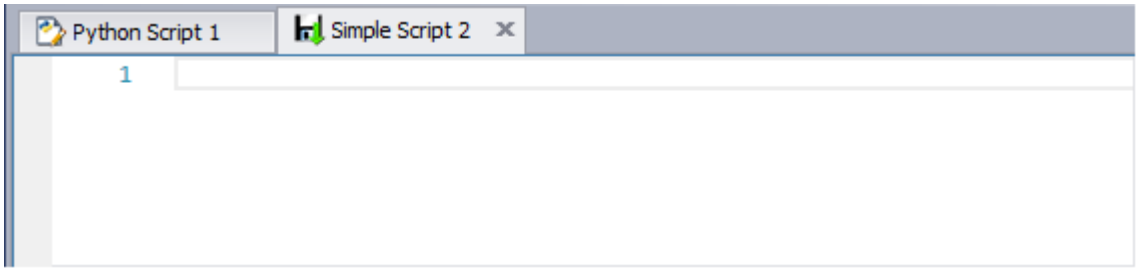
Scripting Examples

Simple Sequential Scripting
Python Scripting

Simple Sequential Scripting

This sample script contains a comment at the top and two SCI commands.

1. Create a new Simple script from the File menu.



Click Run Script to execute the commands.

The commands will be sent to the Velox Message Server and processed. You can see the results in the Command Log window at the bottom of the screen.

Command Log

Start Time	Elapsed Time	Command	Error	Response
9/1/2016 13:23:44.856	00:00:00.014	EchoData "Hello World"	0	"Hello World"
9/1/2016 13:23:44.879	00:00:00.007	ReportSoftwareVersion	0	2.2.0.298

The Event Log window displays information about system events.

Event Log

Time	Level	Message
9/1/2016 11:22:37.058 AM	EVENT	Notification: VMProjectLoaded => 0x170
9/1/2016 10:02:13.898 AM	BEGIN	Register Communication Handler
9/1/2016 10:02:13.914 AM	END	Register Communication Handler
9/1/2016 10:02:12.645 AM	INFO	Reloading Previous Documents
9/1/2016 10:02:13.929 AM	END	Scripting Console loaded and ready
9/1/2016 10:02:11.405 AM	BEGIN	Start Scripting Console

The script file can be saved to a file using the Save As command on the File menu. The file will be saved with the .scr extension.

Python Scripting

Using the Python scripting capability is very similar to Simple Scripting.

1. Create a new Python script from the File menu, and then copy or type the script into the Python Script tab.

Python Script 1.py | new simple script 2.scr

```
1  from velox import *
2  a = 1
3  b = 2
4  c = 3
5
6  response = ReportSoftwareVersion()
7  print(response)
```

2. Click Run Script to display the results in the Output window.

Script Output

```
2.2.0.396
```

Script Output | Variables | Error List | Command Log | Event Log

You can simplify the script (essentially the same functionality as shown in the simple sequential script).

Python Script 2.py | Simple Script 2

```
1  from velox import *
2
3  EchoData('Hello World')
4  ReportSoftwareVersion()
5
```

Because this modified script example does not contain a print statement, the Output window does not show any results. To see command responses, view the Command Log window.

Command Log

Start Time	Elapsed Time	Command	Error	Response
9/1/2016 13:23:44.856	00:00:00.014	EchoData "Hello World"	0	"Hello World"
9/1/2016 13:23:44.879	00:00:00.007	ReportSoftwareVersion	0	2.2.0.298

To use Python (rather than the simple sequential script):

1. Make sure `from velox import *` is included in the first line.
2. Enclose the command parameters in parentheses.
3. Use the # character to start a comment line.

To save the script, click File > Save As. The file will be saved with a .py extension.

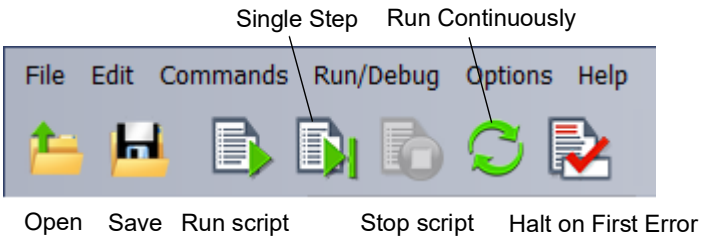
User Interface

For details, see:

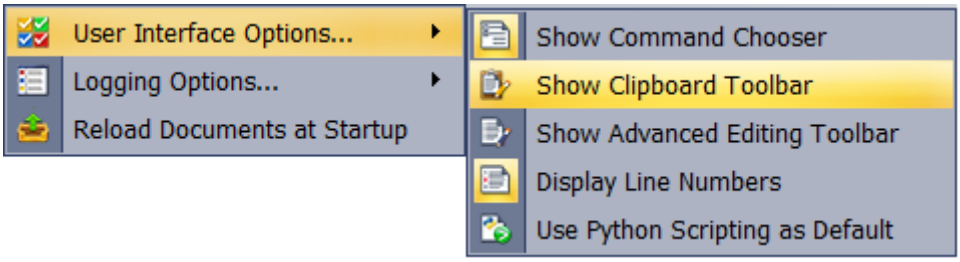
- Toolbars
- Menus
- Status Bar
- Key Shortcuts Quick Reference

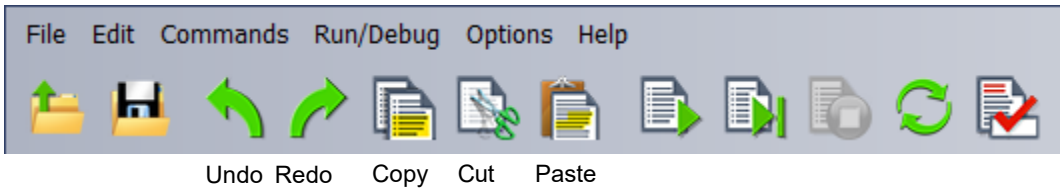
Toolbars

The default toolbar displays the most commonly-used functions. See File Menu and Run/Debug Menu for descriptions of these functions.

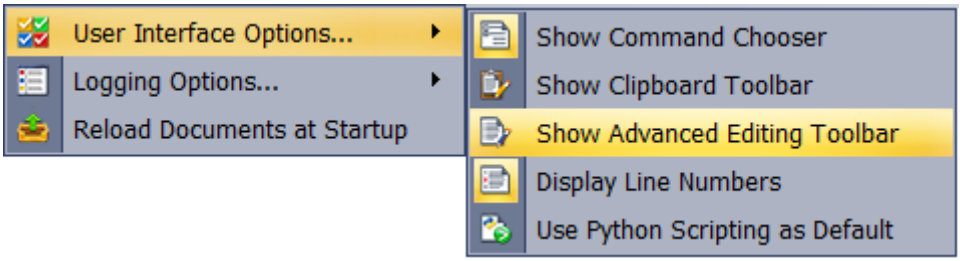


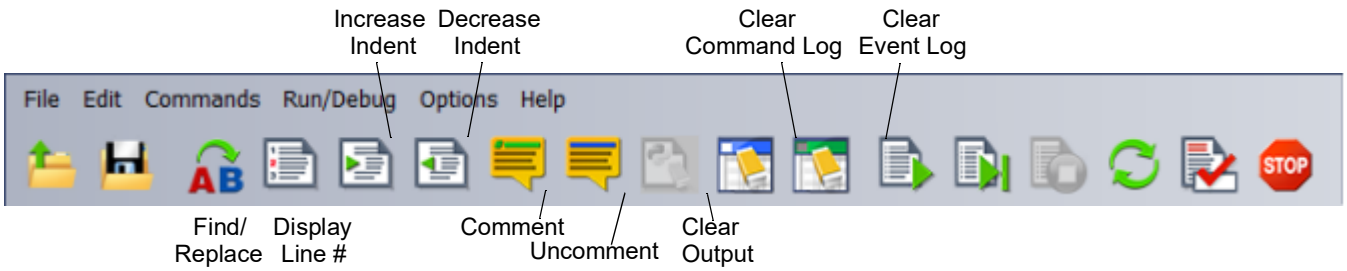
To display the basic editing functions, select Options > User Interface Options > Show Clipboard toolbar. See Edit Menu for descriptions of each function.



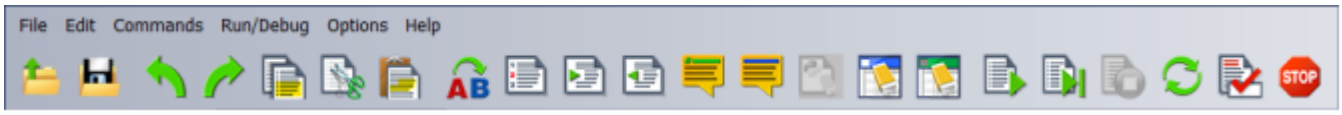


To display the advanced editing functions, select Options > User Interface Options > Show Advanced Editing toolbar. See Edit Menu for descriptions of each function.





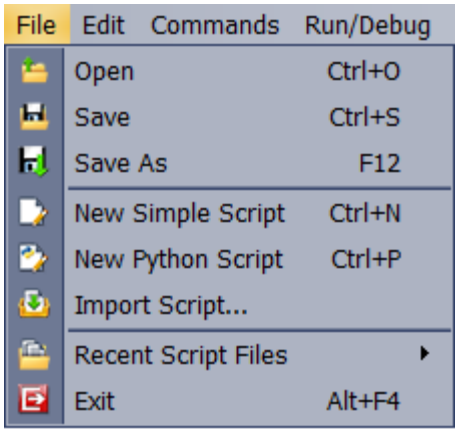
To see all toolbar buttons displayed, enable both Clipboard and Advanced Editing toolbars.

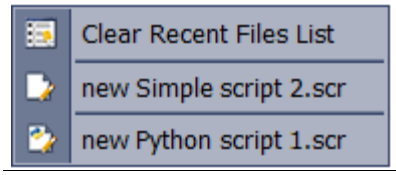


Menus

File Menu
Edit Menu
Commands Menu
Run/Debug Menu
Options Menu
Help Menu

File Menu

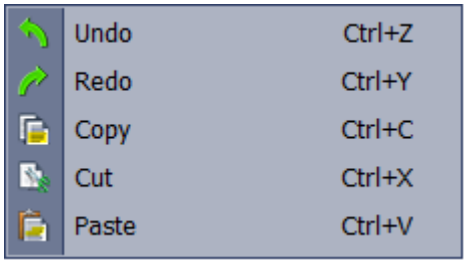




Open	Displays the File Open dialog to open an existing script file.
Save	Saves the current script in an existing file. This item is only enabled if the script file already exists AND the script contents have been changed.
Save As	Displays the Save As dialog to save a script in a new file.
New Simple Script	Creates a new Simple Script in a new editor tab page.
New Python Script	Creates a new Python script in a new editor tab page.
Import Script	• Displays a File Open dialog. • Allows selection of Simple Script .scr files, BASIC .bas files, and Pascal .pas files. • The chosen file will be converted to Python and displayed in a new editor tab page. See Imported Legacy Remote Communicator Scripts.
Recent Script Files	Opens a submenu containing a list the most recently-used script files, and a selection for clearing the list. Clicking on a recently-used script file opens the script for editing.
Exit	Closes the Scripting Console application.

Edit Menu

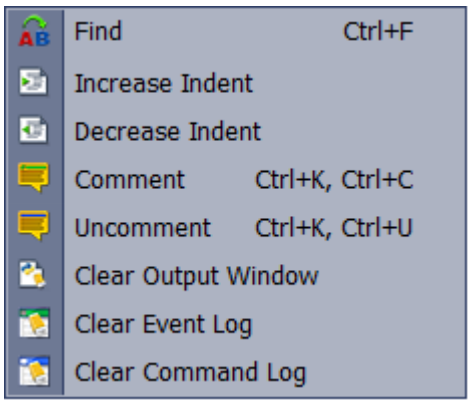
Basic Editing (Clipboard)



Undo	Undoes the results of the most recent editing action. Click multiple times to undo multiple previous actions.
Redo	Reverses the results of the previous Undo operation.
Copy	Copy the selected text to the Windows clipboard.
Cut	Delete the selected text and copy it to the Windows clipboard.
Paste	Insert the text from the Windows clipboard at the current cursor location in the editor.

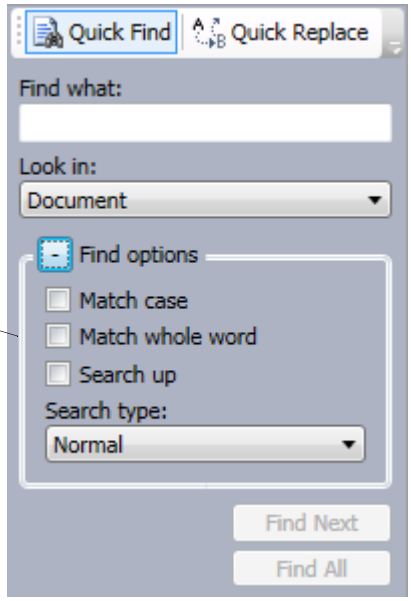
Most of the operations on this menu apply to the text inside the editor pages. The Clear Output and Clear Event Log apply to the output window located below the editor pages.

Advanced Editing



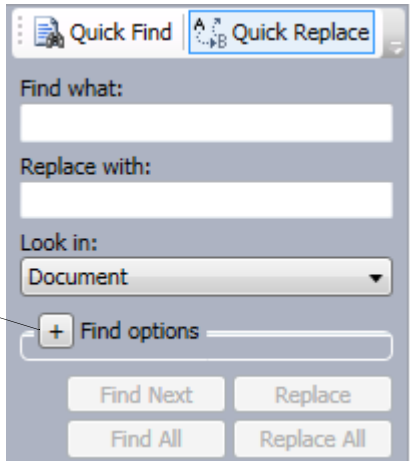
Find/Replace	Opens Find/Replace window at the right side of the editor page.
Increase Indent	Indents the selected text.
Decrease Indent	Decreases the indentation of the selected text.
Comment	Comments out a block of code by prefixing a comment start sequence at the beginning of each line of the selected text.
Uncomment	Removes any comment start sequences at the beginning of each line of the selected text.
Clear Output Window	Clear the contents of the Python output window.
Clear Event Log	Clear the contents of the Event log.
Clear Command Log	Clear the contents of the Command log.

Click to display the Quick Find and Quick Replace tabs.



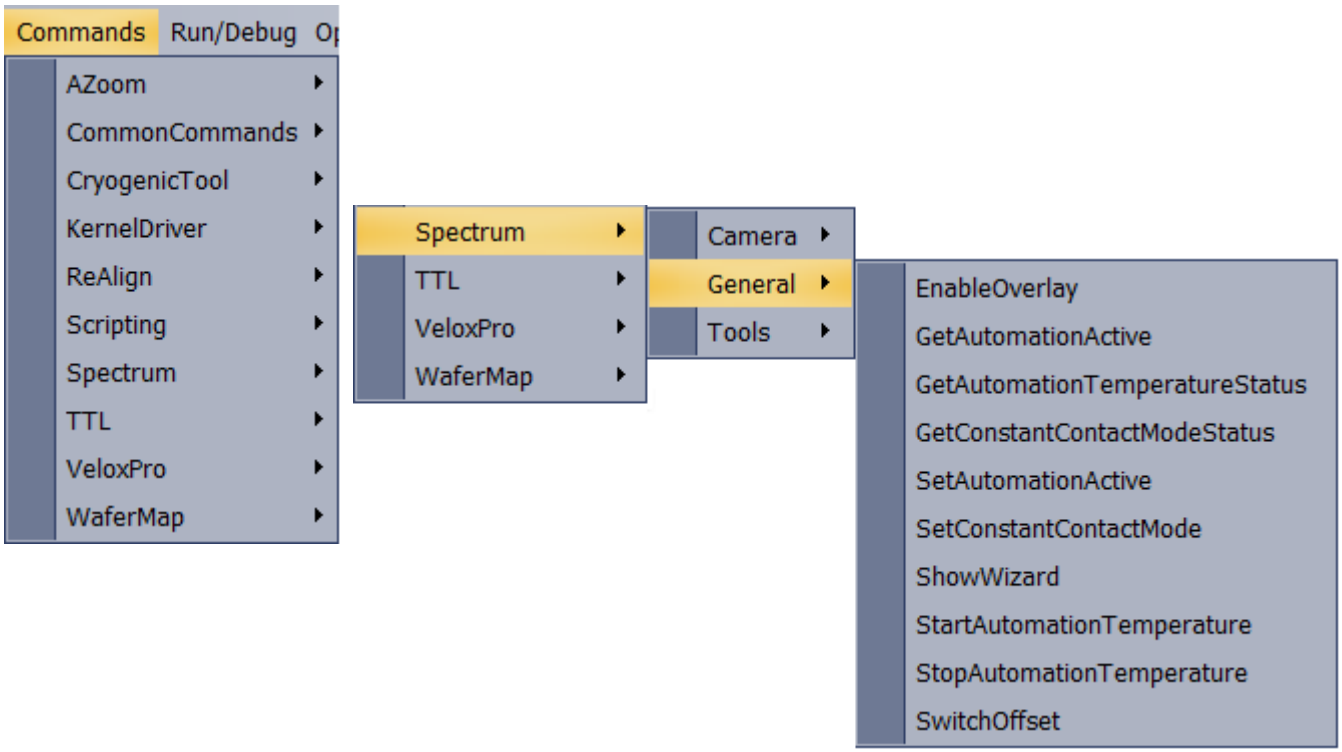
Find options expanded

Click to expand or collapse options



Commands Menu

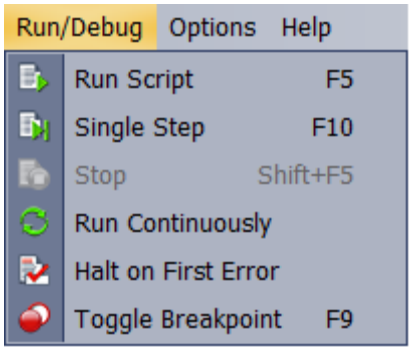
The Commands menu lists command groups available for selection. The example below shows an expanded Spectrum list.

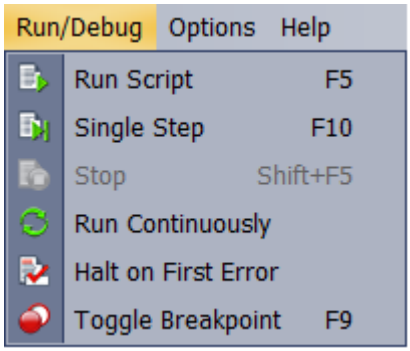


Run/Debug Menu

You can run and debug scripts using these menu selections. The following buttons are disabled in the following circumstances:

- Run is disabled If the script has syntax errors.
- Halt and Stop are disabled if the script is not running.

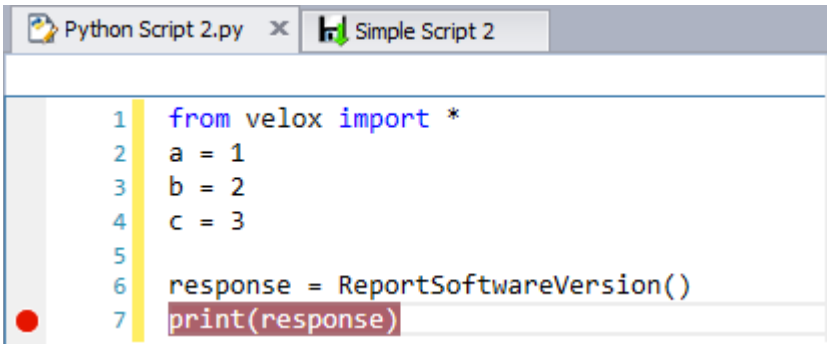




Run Script	Starts running the script on the currently-selected editor page. Disabled if the script is already running.
Single Step	Executes the next command in the script and then pauses. Disabled if the script is running and not paused. Single stepping is useful for debugging.
Stop	Stops the execution of the script displayed in the current editor page. Disabled if the script is not running.
Run Continuously	Starts running the script in the current editor page and repeats continuously until stopped.
Halt on First Error (Simple only)	Stops the script if the SCI command processor reports an error on any command.
Toggle Breakpoint	Sets a breakpoint on the current line in the current editor tab page.

Debugging Velox Scripts

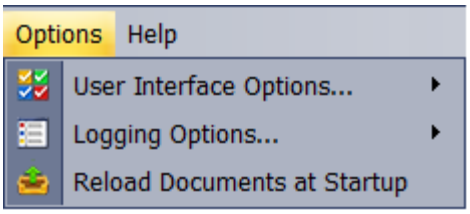
Scripting Console supports breakpoints and single stepping of script files for debugging purposes. To quickly set or clear a breakpoint, enable or disable the Toggle Breakpoint command and then click in the breakpoint margin of the code line. Lines with breakpoints are highlighted in the editor.



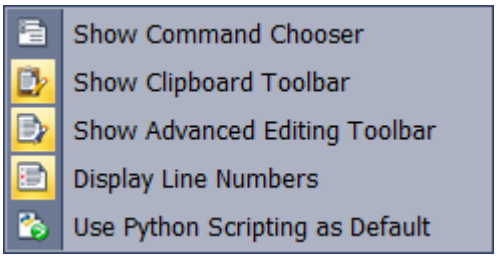
When you click Run Script, the script executes until it encounters the breakpoint. The current line is highlighted in the editor. Click the Run or Single Step button to continue executing the script.

Options Menu

Use the Options menu to specify which toolbars Velox displays and change other settings. Velox saves settings when you close the application and reapplies them when you open the application.

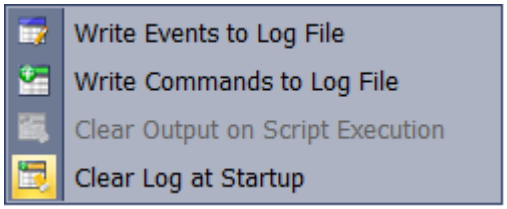


User Interface Options



Show Command Chooser	Toggles the display of Command Chooser at the left side of the window.
Show Clipboard Toolbar	Toggles the display of the Clipboard Toolbar.
Show Advanced Editing Toolbar	Toggles display of the Advanced Editing Toolbar.
Display Line Numbers	Toggles the display of line numbers in the left margin of the editor page.
Use Python Scripting as Default	When selected, the default format for new script files is Python.

Logging Options

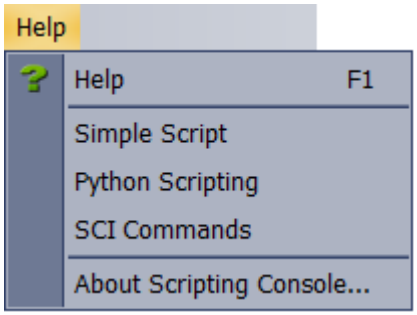


Clear Output on Script Execution	If checked, the Python Output window will be cleared at the start of script execution.
Write Log Entries to File	If checked, all output to the Event Log window will also be written to the Event Log File.
Log System Notifications	If checked, SCI Server and Kernel Notifications will be recorded in the event log.

Reload Documents at Startup

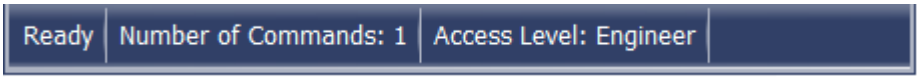
If checked, all documents that were open when the application was closed are reloaded when the application is started. A list of the recently opened script files is also available on the File > Recent Script Files menu.

Help Menu



Help (Shortcut F1)	Display the top level help document.
• Simple Script • Python Scripting • SCI Commands	Click the desired reference. Language References give in-depth information about the scripting language capabilities, syntax, and the available library functions.
About Scripting Console	Displays product, version and copyright information.

Status Bar



Key Shortcuts Quick Reference

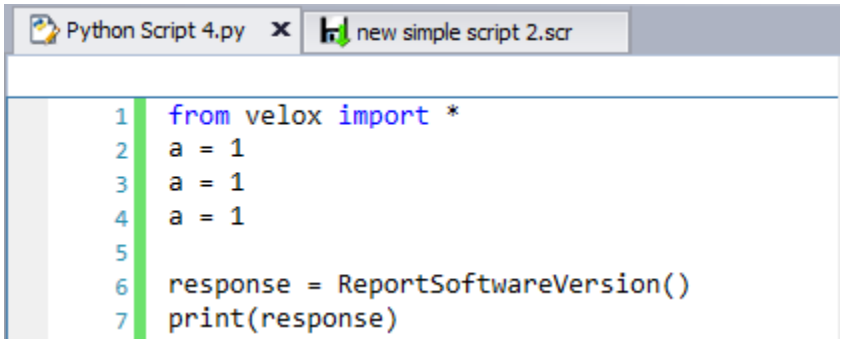
CTRL Key Shortcuts	CTRL-A	Select all text
	CTRL-C	Copy selected text to clipboard
	CTRL-K, CTRL-C	Comment out the selected lines of code
	CTRL-K, CTRL-U	Uncomment the selected lines of code
	CTRL-N	Open a new simple script
	CTRL-O	Open a Script file
	CTRL-P	Open a new Python script
	CTRL-V	Paste the clipboard text at the current location
	CTRL-X	Cut the selected text
	CTRL-Y	Redo
	CTRL-Z	Undo

Function Key Shortcuts	F1	Display the contents of the Help documentation
	ALT-F4	Exit the Scripting Console
	F5	Execute the script in the current editor page
	F6	Check the syntax of the script in the current editor page
	F9	Set or Clear a Breakpoint on the current line
	F10	Single step the debugger
	F12	Save As…

Other Shortcut	ALT-F4	Exit the Scripting Console
	TAB	Increase indentation
	SHIFT-Enter	Execute the current line in simple script.
	SHIFT-TAB	Decrease indentation

Script Editor Window

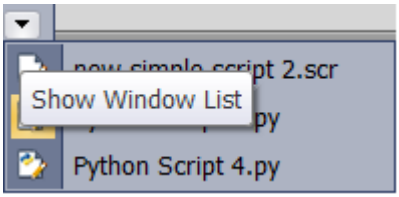
Automation scripts are created and edited in the script editor window. You can edit multiple scripts (both Simple and Python) simultaneously on separate editor pages. Each editor page is represented by a tab in the editor window.



Use the following controls in the Scripting Console to work with scripts:

- New Simple Script or New Python Script on the File menu open new scripts.

- Open File opens existing script files.
- Save or Save As on the File menu or toolbar save script files.
- Scroll right or left to view additional tabs, or click the down arrow to display the list of open scripts.

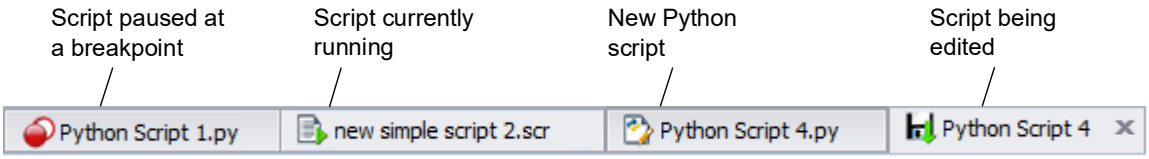


Editor Pages

Editor Breakpoints and Line Numbers

Each tab displays an icon that provides information about the script status. For example, the Run icon appears when a script is currently running.

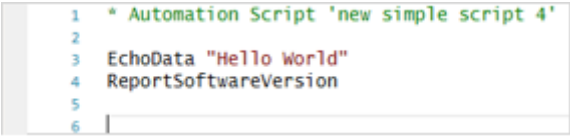
Click the tab of the page you want to view.



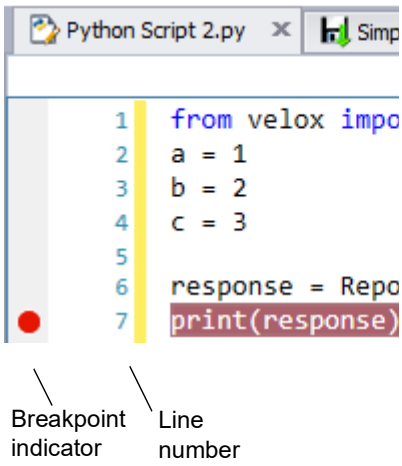
When a script has not been saved to a file, the default name of the script is displayed, as shown in the example above. If the script has been saved to a file (or was opened from a file), the actual file name appears in the tab as shown below.

Color distinguishes the structure of the script. If the script does not display expected element color the code has an unexpected error and must be addressed.

Color		Script language elements
Black	=	Script code
Green	=	Comments
Brown	=	Strings
Blue	=	Script language reserved words



Editor Breakpoints and Line Numbers



The left side of the editor page displays breakpoint indicators as red dots. These dots indicate that the script execution will stop on that line for debugging purposes.

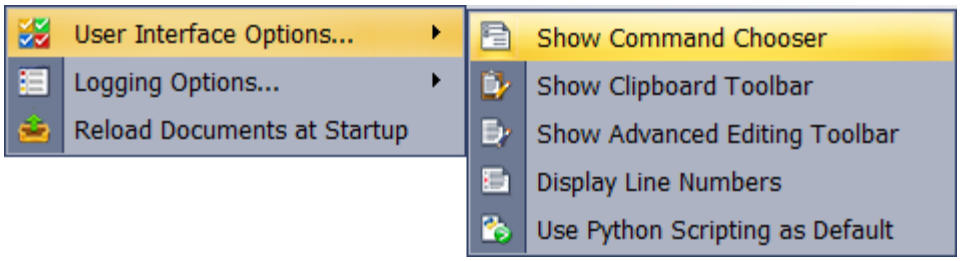
- To set a breakpoint, place the cursor on the desired line of code, then click Run/Debug > Toggle Breakpoint .
- To remove a breakpoint, click in the breakpoint margin of the code line.

Line numbers are displayed in the editor for clarification; they are not part of the script code. To toggle the line number display on (default) or off, click .

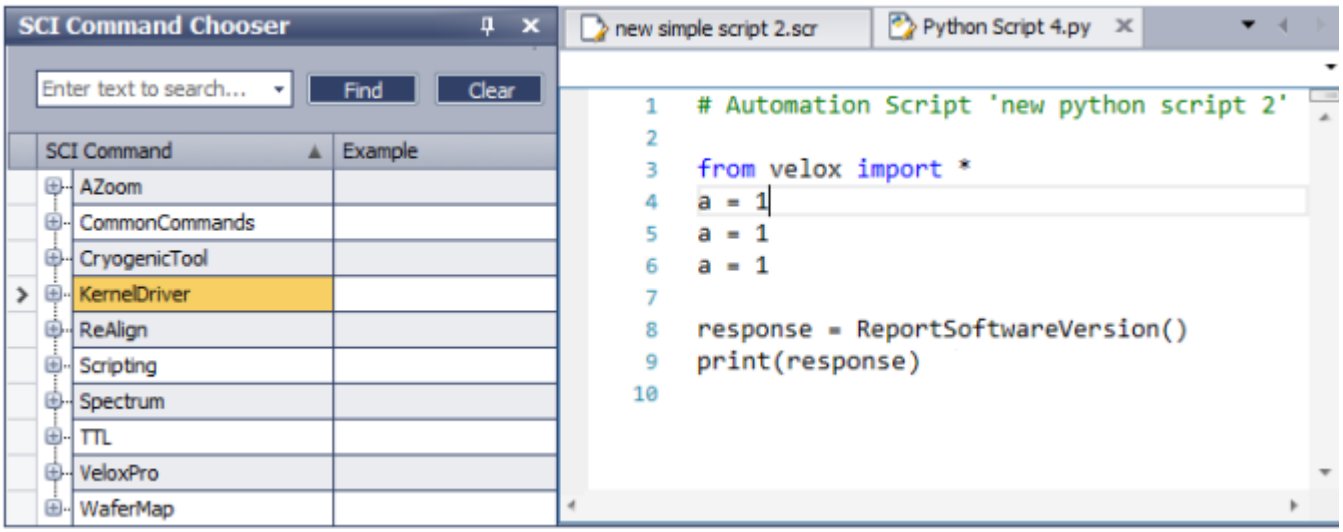
When a Python script error occurs either at runtime or as part of automatic syntax checking, the editor displays the line number of the error. The reported line number corresponds to the line number in the editor.

Command Chooser

Select Show Command Chooser from the Options > User Interface Options menu.

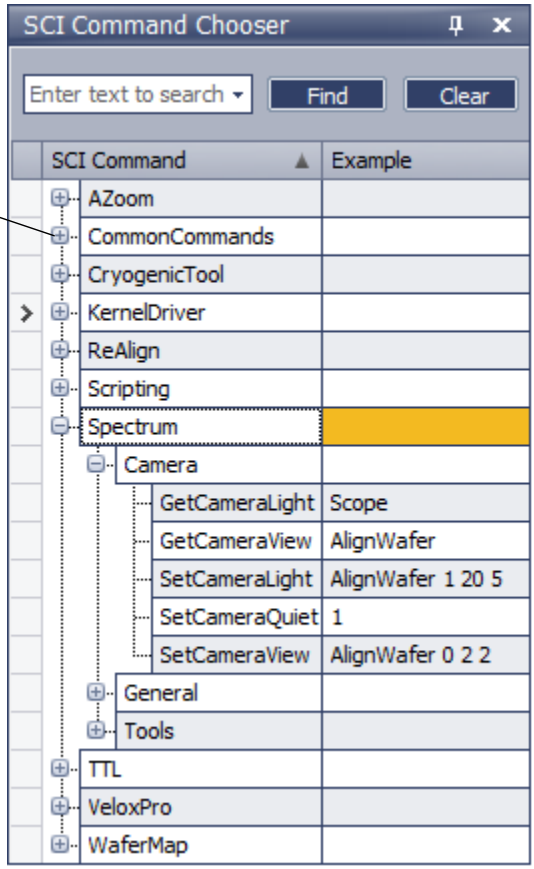


The command chooser displays all of the SCI commands as well as some miscellaneous scripting functions.



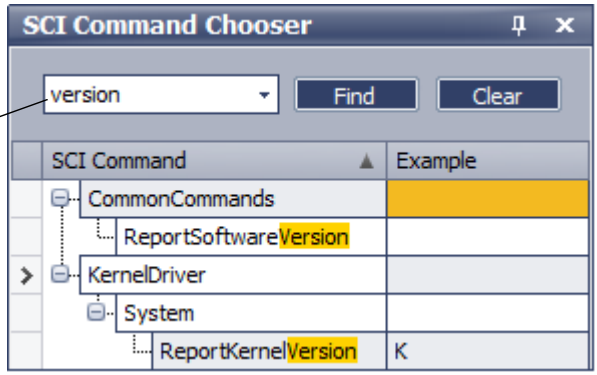
Command Tree

SCI commands are grouped by category in a tree structure. All categories are collapsed when you start the application. Click + on a category or double-click the category name to expand that category and view the available commands.



Command Search

To find related commands, type text in the search bar and then click the Find button. The command tree automatically expands to show the results of the search.



Double Click Insertion	To insert a complete function call at the current cursor location, double-click a command in the tree. The following example shows a function call with sample parameters and a comment specifying the expected inputs and outputs. 23 `LoginDialog("char_1")  # Inputs = (char) Outputs = (none)` For more command details, hold your mouse pointer over an SCI command in the editor.
Drag and Drop Insertion	You can use the mouse to drag commands from the Command Tree and drop them into the editor at a desired location.

Output Windows

The Scripting Console displays output window tabs in the bottom-left corner. Click a tab to show a specific window.

Script Output | Variables | Error List | Command Log | Event Log

The Variables tab applies to Python scripts only.

Command Log

The command log shows the order and timing of every SCI command processed by the script.

- Start Time—When the command was sent to the Message Server, with millisecond resolution.
- Elapsed Time—The amount of time between command sent and response received times.

Command Log

Start Time	Elapsed Time	Command	Error	Response
9/2/2016 12:36:30.947	00:00:00.002	EchoData Hello World	0	Hello
9/2/2016 12:36:30.947	00:00:00.017	ReportSoftwareVersion	0	2.2.0.298

Script Output | Variables | Error List | Command Log | Event Log

- Command—The SCI command string that was sent.
- Error—The error code received in response; 0 indicates no error.
- Response—The response string received.

Event Log

The event log lists various system events, including the beginning and end of script execution, SCI notifications, and so on.

Event Log

Time	Level	Message
9/1/2016 11:22:37.058 AM	EVENT	Notification: VMProjectLoaded => 0x170
9/1/2016 10:02:13.898 AM	BEGIN	Register Communication Handler
9/1/2016 10:02:13.914 AM	END	Register Communication Handler
9/1/2016 10:02:12.645 AM	INFO	Reloading Previous Documents
9/1/2016 10:02:13.929 AM	END	Scripting Console loaded and ready
9/1/2016 10:02:11.405 AM	BEGIN	Start Scripting Console

Error List (Python only) The error list displays current editor syntax errors and runtime errors reported for Python scripts.

Error List

	Line	Severity	Code	Exception	Error Message	Source
>	(8,23)	Error	0		Newline expected.	response = ReportSoftw areVersion()
	(9,4)	Error	0		Newline expected.	pri nt(response.Value)

Variables (Python only) Variables lists the current value of all built-in variables and user-defined variables in the Python script. This list updates each time a breakpoint is hit, when a single step execution is completed. It does not update when you run a script with no breakpoints set.

Variables

	Variable Name	Current Value	Data Type
>	VeloxData	Object	Object
	VeloxLog	Object	Object
	VeloxDataPath	C:\ProgramData\Cascade Microtech\Velox	String
	LastVeloxErrorNumber	0	Int32
	LastVeloxErrorDescrip...		String
	a	1	Int32
	b	2	Int32
	c	3	Int32
	response	('2.2.0.396',)	PythonTuple

This list is empty when the script isn't running.

Script Output (Python only) This window displays the output from script `print` statements.

Script Output

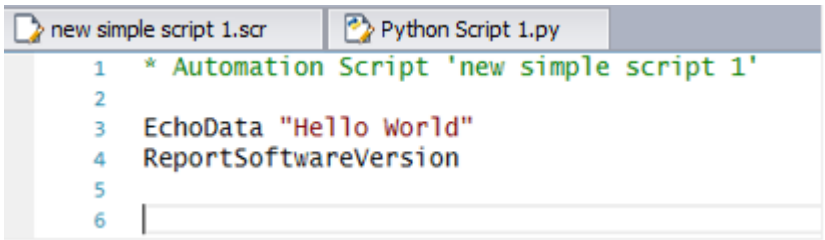
2.2.0.396

Simple Sequential Scripting Language

Simple Sequential Scripting

Simple scripts execute line by line from top to bottom with no branching or looping. Simple script does not use variables or arithmetic expressions.

Each line of a simple script is an SCI command or a comment.

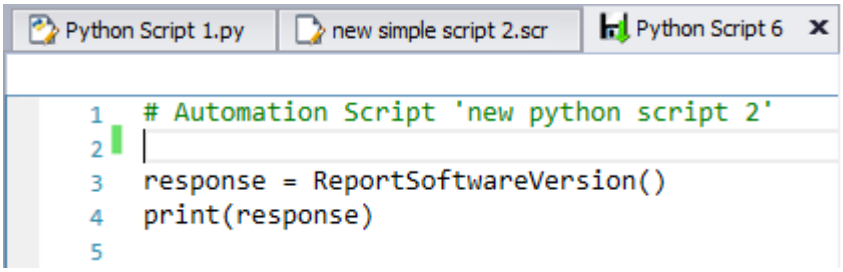


"Hello World" in Simple Script	A minimal simple script can be a single line with just one SCI command. A very simple example is the "EchoData" SCI command, which sends the string provided as a parameter to the Message Server; the Message server sends it back as a response. Click the New Simple Script button on the File menu to generate a new script.
Viewing SCI Command Responses	The *Command Log* lists responses to SCI commands sent. Click the Command Log tab to display the window if it is not already shown.
Descriptive Comments	Simple Script supports descriptive comments in the code to describe what the script does. They have no impact on the script code executed. Comment lines are not sent to the message server or executed in any way. A simple script comment begins with the character `*` followed by a space. Everything that follows on that line is ignored. The editor highlights comments in green.
SCI Commands	SCI command form is the SCI command name followed by a space and then the parameter values, separated by spaces. `CommandName parameter1 parameter2 …`
Using the Command Chooser	*Command Chooser* provides quick access to all SCI commands. To add the command, any example parameters, and a comment about inputs and outputs to the script, double-click a command name or drag and drop it into the editor.
Special Functions	A limited number of special functions available in Simple Script are preceded by an asterisk with no spaces. `*Delay` is an example of a special function. The *Delay function pauses execution for the specified number of milliseconds before proceeding with the next command. `Function: *Delay ms`
Things to Remember About Simple Script	• Comments start with an asterisk followed by a space and continue until the end of a line. • Parameters are separated by spaces. • Simple Script does not check for syntax errors prior to execution. • Only the Command Log window and Event Log window are visible.

See also Scripting Examples.

Python Scripting Language

Python is an application development programming language that supports large, complex programs, extensive function and class libraries, and a rich editing and debugging environment. It is also well-suited to simple scripting applications. The Python environment, supported by the scripting application, supports all of the features of the Python 2.77 specification. For more information, see the IronPython 2.7 documentation at https://python.org.



For more information, see:

- Python Overview
- SCI Commands
- Miscellaneous Scripting Commands
- Symbol Navigation
- Introduction to Python Programming

See also Scripting Examples.

Python Overview

Python is a widely supported programming language. Many books, tutorials, courses, and Internet references are readily available. Most code samples found on the Internet will run in the scripting application.

Simple Sequential Scripting in Python	Creating a simple sequential script (one with no branching or looping) in Python is very similar to a simple script. The Python script needs just one extra line of code to specify the Python SCI library module. Similar to most other programming languages, Python functions enclose parameter lists in parentheses and separate the parameters with commas. Variables and other program structures not available in Simple Script are also easy to use.

"Hello World" in Python	The simple script Hello World example shown looks very similar in Python script.
	`from velox import *`
	`"EchoData World')`
	`ReportSoftwareVersion()`
	• Comments start with a # character.
	• The `from Velox import *` statement is required to support SCI commands.
	• Function parameters are enclosed in ( ).

Descriptive Comments	Comments begin with the # character and continue to the end of the current line.
Python Flow Control and Code Indentation	Unlike many programming languages, indentation of code is important in Python. Where other languages might use begin and end markers for code blocks, Python relies on formatting. For this reason, all statements in a simple Python script should start in the first column. In Python, an IF statement starts with the keyword `if`, followed by a conditional test, and ends with a colon. The next line of code should be indented to specify that code will be executed if the condition is true. All indented statements are executed. When the indentation is decreased, the code block is complete.

This example script fragment illustrates how Python uses indentation for code structure instead of begin and end markers such as { and } as in C# or C++.

```
11  if b > a :
12      print(True)
13      print('true again')
14
15  print('continue')
```

- Line 11: The keyword `if` is followed by the conditional `b > a` and ends with a colon.
- Line 12: A print statement, indented one level, will beexecuted if b is greater than a.
- Line 13: Another print statement will be executed if b is greater than a because it is also indented.
- Line 15: A print statement that will be executed regardless of the if statement because it is not indented at the same level.

The output from this code looks like this if b is > a:

```
Script Output
True
true again
continue
```

Because the if statement was true, the indented code was executed.

When b is not greater than a, the output will look like this:

```
Script Output
continue
```

In this case, the indented print statements were not executed because the if statement was not true.

About Python Library Modules

Python uses function libraries called modules to add capabilities to the language. The standard Python library includes hundreds of functions. Modules are text files with a .py extension or a compiled module with a .pyc file extension.

Do not include the .py extension when you use an import statement.

- The SCI commands are included in a module called *velox.py*.
- Legacy support for converted Pascal and BASIC scripts is provided by the *vxlegacy.py* module.
- VISA device communication (GPIB and RS-232) is provided by the *vxvisa.py* module.

Scripts need only import the modules that will be used by that script. Almost all scripts should import the *velox* module for SCI command support.

Error Indications

A red underline highlights detected errors. Position the cursor over the highlighted text to display the error message.

```
7
8   resp onse = ReportSoftwareV
9   print(response[0])
10
```

Errors are listed in the errors panel below the editor.

Things to Remember About Python

- Comments start with a # and continue to the end of a line.
- Import statements are required to access some function modules.
- Python supports IntelliSense code completion.
- Most flow control statements require a colon at the end of the line.
- Code indentation is part of the language structure and denotes a code block.
- Python uses lowercase letters for most names and statements.
- Variables do not need to be declared before they are used in Python.

Commands

SCI Commands
Miscellaneous Scripting Commands
User-Defined SCI Commands in Python Script

SCI Commands

Each available SCI command is represented by a Python function call in the supplied *velox.py* library module. The function name is exactly the same as the SCI command name.

Calling SCI Command Functions	SCI command functions are called as any other function. Give the function name followed by a list of parameters enclosed in parentheses. If no parameters are given, use an empty set of parentheses.
	`ReportSoftwareVersion()`
	
	`SelectAZoomLens(1)`
	
	`OCLightVal(128,1,4)`
	
	`AddSubDie (x,y,label)`
	This example has four SCI command function calls. The comments and example parameters were automatically added by the Command Chooser.
	• ReportSoftwareVersion() takes no parameters.
	• SelectAZoomLens(1) take a single integer parameter.
	• OCLightVal(128,1,4) takes 3 integer parameters.
	• AddSubDie takes two decimal values and a string.

SCI Command Reponses	SCI commands may return one or more values. Each value is returned as a separate variable to your script. The three ways to access the return values are as follows:
	
	Return Values as Variables – Python syntax allows for multiple return values, separated by commas. Call the functions like this:
	`ver, desc = velox.ReportKernelVersion()`
	`print('Got ', ver, 'and', desc)`
	
	One Return Value – If only a single is expected, call the function like this:
	`evalue = velox.EchoData("Echo_This")`
	
	Return Values as Python namedtuples – Python syntax allows for multiple return values as a named group returned as an object. To reference the return values this way, call the function like this:
	`v = velox.ReportKernelVersion()`
	`print('The kernel version is', v.Version, 'and', v.Description)`
	`print('The kernel version array is', v[0], 'and', v[1])`
	
	You can also access the return values as an indexed array.

Optional Parameters and Named Parameters	Most parameters to SCI commands are optional. However, if parameters are given, they must be in the specified order. Only parameters at the end of the list may be omitted. If a parameter is provided: • All parameters preceding it must also be provided. • Parameters following it can be omitted. Using the function `AddSubDie (x,y,label)` as an example, if the second parameter (`y`) is to be included, the first parameter (`x`) must be supplied as well, but all of the remaining parameters (`label`) may be omitted.

Miscellaneous Scripting Commands

In addition to the SCI commands, a set of additional functions specific to Velox is supported.

```
from velox import *

VeloxLog.off()
VeloxLog.on()
print(VeloxData.lastErrorDescription)
print(VeloxData.lastErrorNumber)
VeloxLog.writeLine("Test Velox Log")# VeloxLog.writeLine(string)
```

VeloxLog

VeloxData.lastErrorNumber returns the error code for the most recent SCI command executed.

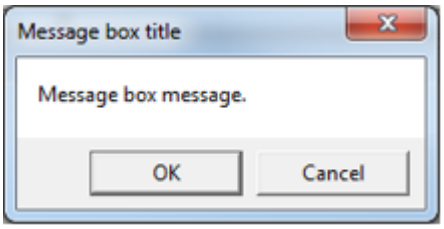
Event Log

	Time	Level	Message
	1/4/2017 08:51:21.560 AM	BEGIN	Execution of "Python Script 1.py" started.
	1/4/2017 08:51:21.748 AM	ERROR	Execution of "Python Script 1.py" terminated with error.
	1/4/2017 08:51:35.108 AM	BEGIN	Execute Script File: Python Script 1.py
	1/4/2017 08:51:35.118 AM	BEGIN	Execution of "Python Script 1.py" started.
	1/4/2017 08:51:35.318 AM	END	Execution of "Python Script 1.py" completed normally.
	1/4/2017 08:51:43.364 AM	BEGIN	Execution of "Python Script 1.py" started.
	1/4/2017 08:51:43.422 AM	INFO	Execution of python script "Python Script 1.py" PAUSED.
	1/4/2017 08:51:45.994 AM	INFO	Execution of python script "Python Script 1.py" RESUMED.
	1/4/2017 08:51:46.134 AM	INFO	Execution of python script "Python Script 1.py" PAUSED.

`VeloxLog.on()` enables writing to the message log file.

`VeloxLog.writeline(string)` writes a line of text to the message log.

`VeloxLog.off()` disables writing to the message log file.

ShowMessageBox	The `ShowMessageBox` command allows the script to display a popup message box and receive a user response based on a button click. The function call has this form:
	`result - ShowMessageBox('Message box title', 'Message box message.', MessageBoxStyles.OKCancel`
	
	`print(result)`
	
	The return value is an integer representing the number of the button that was clicked.
	Valid button numbers are:
	`OK = 1, Cancel = 2, Abort = 3, Retry = 4, Ignore = 5, Yes = 6, No = 7, TryAgain = 8, Continue = 9`
	The message box displays the title at the top and the text in the body.
	 
	
	`MessageBoxStyle` specifies the buttons that appear on the message box. Values for the style may be:
	`OKCancel, AbortRetryIgnore, YesNoCancel, YesNo, RetryNo, CancelTryAgainContinue`

User-Defined SCI Commands in Python Script

User-defined SCI commands are not part of the *velox.py* module imported for Python scripting.

For example, if you have a command called `myCustomUserCommand` defined in the *commands.user.txt* file (see *Velox Integration Toolkit Guide* on the installation DVD), this is not a recognized Python command such as `ReportSoftwareVersion` or other standard SCI commands.

To send a custom SCI command, use the Python velox function, `SendSciCommand`. This function takes the command name and a string, followed by a list of parameters. For example:

```
from velox import *
results = SendSciCommand("myCustomUserCommand", x, y, z)
```

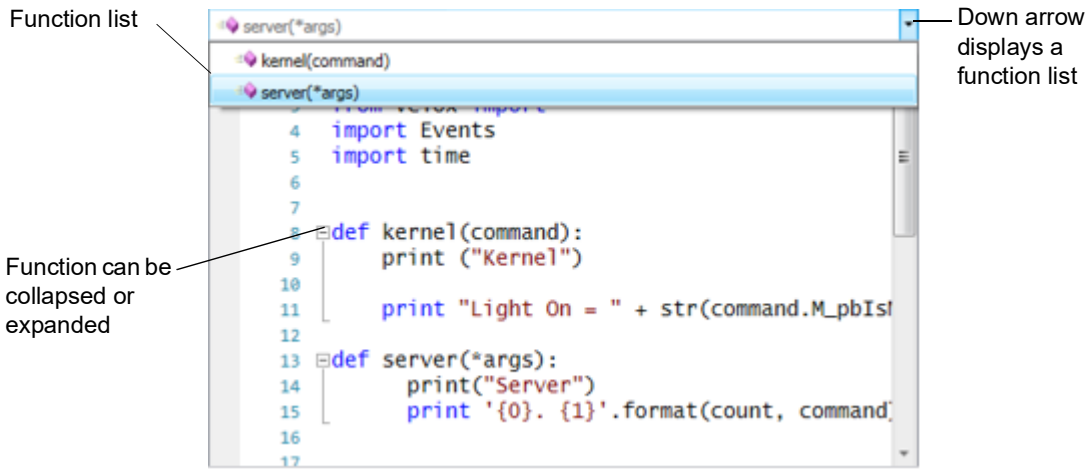
It is also possible to create a Python module to give full intellisense capabilities. Create a module like *myscicommands.py* and create a simple function for each command. For example:

```
from velox import *

def customCommand1(x, y, z):
    return SendSciCommand("customCommand1", x, y, z)

def customCommand2():
    SendSciCommand("customCommand1")

result = customCommand1(x, y, z)
customCommand2()
```

This requires just two lines of code for each custom command. In a case where no values are returned by the command, the `return` keyword is not needed. Collect these functions in a .py file and import it as a library in order to use them like any other Python function.

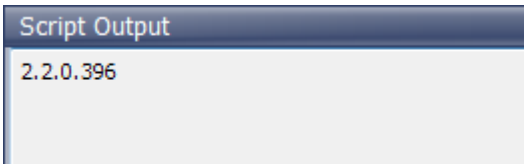
Symbol Navigation

Click the down arrow at right to view the list of all functions currently defined in the script file.



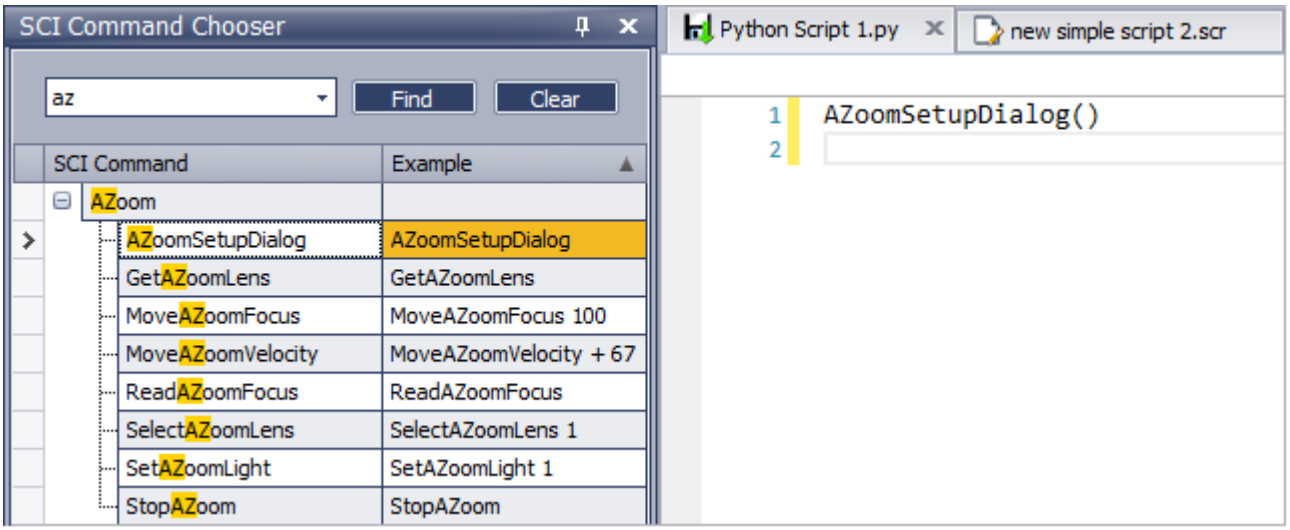
Script Output

The Script Output window displays the output from script `print` statements. The dynamic contents of the window reflect the script currently selected in the editor.



IntelliSense and Automatic Code Completion

Intelligent code completion, or IntelliSense, reduces the amount of typing and API lookup required to write script code. In the following example, the characters `az` are entered in the SCI Command Chooser, which automatically lists all the SCI commands and library functions associated with those characters.



You can use the mouse pointer or the arrow keys on the keyboard to select or examine the commands and examples. To insert the name into the editor at the current cursor location, press the Tab key, press the Return key, or double-click the function name. Press the Esc key to exit IntelliSense help.

To view the definition of the function and the parameters that the function supports, hold the mouse pointer over the function in the editor or type "`(`" after the function name. Most parameters for SCI commands are optional.

This shows all of the parameters that the function expects and the type of data to be used. As parameters are added, the editor highlights the next expected value in the pop-up. Brackets `[]` indicate optional parameters.

Introduction to Python Programming

A brief introduction to the Python programming language will help you quickly get started writing functional scripts.

Variables	To use a variable, assign a value to a name.
	`a = 1`
	`b = 2`
	`c = a+b+3`
	• Variables do not need to be declared before they used.
	• Variables do not need to declare the type of the variable (integer, string, etc.).

IF Statements	Notice that the `if` and the `else` lines both end with a colon, and the code to be executed in each block is indented.
	`if a < b:`
	`  print("a < b")`
	`else:`
	`  print("a is not less than b")`

FOR Loops	The range of a `for` statement goes from the first number in the range up to one less than the second number in the range. In the preceding example, the example below prints values 1-99.
	`for x in range(1,100):`
	`  print(x)`

Function Definitions	• Define a function using the `def` keyword.
	• Give the function a name and a set of input parameters.
	`def printRange(start, end):`
	`  for x in range(1,100):`
	`  print(x)`
	`  return end -1`
	
	`lastNumber=printRange(1,100)`
	`print("last="+str(lastNumber))`
	
	• The def statement ends with a colon, the same as most Python statements that use code blocks.
	• Indent the code to be executed.
	• If a value is returned by the function, use a `return` statement followed by the expression to be returned.
	• Call the function as shown on line 18.

Learning More About Python Programming	This document gives a quick introduction to Python for use scripting SCI commands. It is not intended to thoroughly teach Python programming. For more information, see python.org. Velox installation includes sample Python scripts located in C:\ProgramData\Cascade Microtech\Velox\Scripting Console\Scripts\Python. Search the Internet for Python programming tutorials.
Adding 3rd Party Python Libraries	The Scripting Console supports adding most third-party Python libraries that are compatible with Python 2.7. The common method for adding third-party libraries is the Python pip command. If installed, Scripting Console uses modules and packages located in the standard Python 2.7 folder. To use third-party libraries: • Create the proper folders and files needed to use third-party packages. – Install the standard Python 2.7 runtime for Windows (https://www.python.org/downloads/). – Add the Python 2.7 search path ";C:\python27" to Windows Path environment variable • Add a package: – Open a command prompt window (with administrator privileges) – Type the following: > `CD \Python27` > `Pip install packagename` (where `packagename` is the package to install) • In the script file, add an import packagename statement.

Imported Legacy Remote Communicator Scripts

You can import and convert .scr, .bas, and .pas scripts from Velox Remote Communicator to the Scripting Console syntax automatically. When you import one of these script files, it is translated to a Python language script.

To import a script file, click File > Import Script on the Scripting Console menu. A file explorer window opens for you to choose a file to import.

The file you choose is translated to the Python programming language, and the Scripting Console displays it in a new editor page as a Python script. The name of the script is the same as the original file name until you save it with a new name.

Although most syntax of the legacy Remote Communicator files is converted correctly, sometimes the importer does not handle certain language features, such as:

- Comments may appear on a separate line instead of at the end of a line.
- Variant arrays are not translated.
- Some library functions are not translated.

Some user scripts must be edited by hand after importing.

Test your script after you import it. Any library functions that have not been translated appear as errors at run time. You can substitute equivalent functions from the Python library. If the importer can't add a statement, it adds an error comment to the end of the script.

Be sure to save the file after the import and any hand editing are completed.

File Conversion

Imported files are automatically converted to Python scripts. Following is more information on .scr files, .bas files, and other information about legacy file import.

Simple Script .SCR Files

You can execute simple script (.scr) files without modification from the Scripting Console. One line of the .scr file is translated at a time.

Element	Simple Script Before	Python Script After	Notes
Comments	* my comment	# my comment	
Blank Lines			Unchanged
SCI Commands	SelectAZoomLens 1	SelectAZoomLens(1)	Changed to function call
SCI Post Commands	*SelectAZoomLens 1	SelectAZoomLens(1, post=true)	Changed to post function call

Basic Script .BAS Files

Python does not require variable declarations or type declarations. DIM statements without value assignments are removed.

Python does not have a Select/Case statement. These are translated to If/Else If statements.

End statements are removed.

Some empty statements are translated to Python Pass statements.

File Operations

File operations are converted to native Python functions. This often results in multiple redundant calls to the open() file function. The translated code should work as intended, but the code can often be optimized with very little effort. In many cases, redundant calls to open() can simply be deleted.

File operations are often used for logging messages to a file. Built-in Python logging functions may be easier to use. See the sample scripts for Python Logging Operations. Velox also provides a set of logging APIs through the VeloxLog Python functions.

Optimizing Imported Scripts

The legacy script importer attempts to correctly translate BASIC and Pascal scripts into Python. However, the code will be a direct translation of the original script logic and will not be optimized

for he Python language. Most scripts may benefit from simple hand optimization of the translated code.

Comment Lines

Most comments in existing scripts will be preserved during import translation. However, the placement of the comment may be different than in the original code or, in some cases, missing.

Because a comment line may appear at any point in a script, there are situations where the importer is not able to preserve the comment.

When a comment is placed on a separate line, there may be times where the comment gets appended to the previous line in the translated code. When the comment appears at the end of a line, there are times when it may be moved to a separate line.

Library Functions and Limitations

Many, but not all library functions from the BASIC and Pascal scripts are implemented. These functions are provided only for automatic translation of BASIC and Pascal into Python. They should not be used when writing new Python scripts. Use native Python functions when possible.

Functions which are specifically not implemented include the Variant Array functions:

- VarArrayCreate
- VarArrayHighBound
- VarArrayLowBound

The variant array is supported by Python, but an automatic translation is not provided.

Legacy commands supported by Scripting Console:

Other	Math	Strings	Date / Time	File
DataPath	Abs	Chr	Date	Append
Delay	ArcTan	IntToHex	DecodeTime	Assigned
GetErrorDescription	Cos	IntToStr	IsLeapYear	AssignFile
LastError	Dec	FloatToStr	Now	CloseFile
LogErase	Exp	FormatFloat	Time	FilePos
LogOff	Inc	Int	DateToStr	ReadLn
LogOn	Ln	Length	DateTimeToStr	Reset
VarIsNull	Round	LowerCase	TimeToStr	Rewrite
ChDir	Sin	StrToFloat	DayOfWeek	Write
Ord	Sqr	StrToInt	EncodeTime	WriteLn
	Random	Trim	EncodeDate	FileSize
	Trunc	TrimLeft	DecodeDate	
	Frac	TrimRight		
	Odd	UpperCase		
		VarToStr		
		Pos		
		Copy		
		Insert		
		Delete		
		AnsiCompareStr		
		AnsiCompareText		
		AnsiLowerCase		
		AnsiUpperCase		
		CompareStr		
		CompareText		

GPIB Serial Communications

Communicating with an instrument over RS-232 or GPIB takes just a few lines of code. Below is a minimal script that opens a serial port and reads a response from a message over RS-232.

```
from velox import *
from vxvisa import *

azoom = Instrument() # create an instance of the instrument class
azoom.open(azoom.ports[0]) # open a port
response = azoom.query("QZ")
```

All that is required is that the script does these steps:

1. Create a variable from the Instrument class
2. Open the port using the ".open" method

This example uses the port as currently-configured by the computer settings. The system uses the National Instruments VISA libraries for communication. See the VISA documentation for a complete reference.

Configuring the Interfaces

Configuring GPIB

The GPIB interface should be configured through a separate application prior to opening the device in Scripting Console. Scripting Console will use the device settings specified for the interface.

Configuring Serial Ports

The Serial interface should be configured through the Windows Control Panel application prior to opening the device in Scripting Console. Scripting Console will use the device settings specified for the COM Port interface.

Error Handling

Timeouts and other errors should be handled by a try/catch block in your script.

```
# Automation Script 'new script 2'

from velox import *

azoom = Instrument()       # create an instance of the instrument class
print azoom.ports          # look at the list of available ports
azoom.open(azoom.ports[0]) # open a port
print azoom.protocol       # look at the protocol
print azoom.portName       # look at the port name

try:
    response = azoom.query("QZ")
    print "Azoom says: {}".format(response)
except:
     print "Exception Caught"
```

WinCal Remoting with Scripting Console and Python Script

It is possible to remotely interact with WinCal through the WinCal remoting APIs. These APIs are provided on the WinCal installation media as a separate install package. To use WinCal remoting, first install both WinCal and the WinCal remoting application.

WinCal remoting uses a facility of Windows called '.NET remoting'. This allows .NET applications, written in Visual Basic.NET or C#, to interact with the WinCal application. It is possible to use the WinCal remoting APIs in Python scripts written in the Velox Scripting Console because of the integration with .NET available from the version of Iron Python supported by Velox Scripting Console.

Note that this example will not work with a generic installation of Python on Windows or Linux. Other Python support for calling Windows DLLs may exist, but are not discussed here.

Using WinCal Remoting from Scripting Console

There are three steps to interaction with WinCal from Python script: import the API modules, connect to WinCal server, and call the APIs needed by your application.

Note that WinCal remoting is not available in Simple Script.

Importing WinCal Remoting Modules

See WinCal documentation for information about specific APIs available.

Importing the WinCal Remoting DLL requires three lines of code in your Python script: import the 'clr' module, add a reference to the DLL, and import the DLL with a name that is easy to use.

```
import clr
clr.AddReference('WinCalRemoting')
import CMI.WinCalRemoting as wincal
```

The code above makes the WinCalRemoting APIs available for use in your Python script. Now your script can call WinCal.

Because CMI.WinCalRemoting is a Windows DLL, not a Python module, the 'IntelliSense' features of the Scripting Console editor will not be available for WinCal APIs. See the WinCal User Guide for a list of available APIs. The API names are case sensitive and must match exactly. If you get an error saying that a function is not found, check the spelling and capitalization.

Connecting to the WinCal Server

Before calling APIs to interact with WInCal, the script must create a connection to WinCal. An example Python fragment is shown.

```
w = wincal.cWinCalClient()
success = w.WinCalOpenServer('localhost', 22778)
print(success)
```

Calling WinCal APIs From Python Script

After creating the client reference and connecting to the server, make API calls according to your needs. A complete example is shown below.

```
# import the dll and add a reference
import clr
clr.AddReference('WinCalRemoting')
import CMI.WinCalRemoting as wincal

# Create the client. This example uses a variable named w
```

```
# as the reference to the WinCal Client. After connecting
# to the server, all API calls will be referenced as members
# of w.

w = wincal.cWinCalClient()

# This example uses 'localhost' to indicate that WinCal
# is running on the same machine as the script.
# If the WinCal machine is remote, use the target machine
# name or IP Address. The second argument is the socket #

success = w.WinCalOpenServer('localhost', 22778)
print(success)

# The client is attached to the server and we can make API calls.

#Get the name and version number of the WinCal server that we attached
to
print(w.WinCalServerName())
print(w.WinCalServerVersion())

# Open the WinCal event window and display a custom message
w.EventWindowShow();
w.EventWindowAddMsg('Hello WinCal')
```

Chapter

7 Server Commands

The following command sets and notifications are included in this section:

- Common Commands
- Notifications

Common Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands		
ExecuteCleaningSequence	GetStatus	ReportSoftwareVersion
GetAlignmentMode	IsAppRegistered	SaveProject
GetLoginData	LoginDialog	SaveProjectAsDialog
GetProjectFile	OpenProject	SetAlignmentMode
GetSoftwarePath	OpenProjectDialog	

ExecuteCleaningSequence

Description

Executes the specified cleaning sequence as defined by the ControlCenter Cleaning Manager. If no sequence is specified, the default sequence is used. If no default sequence is specified, it will clean (CleanProbeTip) using the first cleaning aux site and then move to the first verify site and make contact.

ExecuteCleaningSequence checks that there is enough cleaning media available before performing the cleaning. If not enough media is available, it will return an error immediately. The tester can send the ResetCleaningPosition command if it wants to continue using the same cleaning pad.

Parameter Input

SequenceName (optional)

The sequence name.

Default ""

AllowMediaReuse (optional)

A value of 1 will force cleaning when the cleaning media is used up. It will start over on the same media. If 0, an error is returned if there is not enough media left to complete the sequence.

Default: 0

SkipAlignAux (optional)

A value of 1 will skip the Spectrum AlignAux alignment for clean sites that have a defined cleaning reference. Default: 0

SkipReturnMove

Skip return move back from cleaning site. Default: 0

Example

```
ExecuteCleaningSequence OnWaferDeepClean
ExecuteCleaningSequence "" 1
ExecuteCleaningSequence DeepClean 1 1
ExecuteCleaningSequence
```

GetAlignmentMode

Description

Gets the active alignment mode (either on axis or off axis).

Parameter Output

AlignmentMode

0 – Off Axis

1 – On Axis

Example

```
GetAlignmentMode
```

GetLoginData

Description

Gets the Velox user data: name, full name, group, access level. Will return an error if the user is not set up with a login account.

Parameter Input

CmdUserName

Short identifier of the user

Parameter Output

User name short

Short identifier of the user

User name long

Full user name

Group name

Full name of the user group for this user

Access level

Access level associated with this user

Velox locked

Example

```
GetLoginData
```

GetProjectFile

Description

Returns the current project file.

Parameter Output

File Name

Example

```
GetProjectFile
C:\Users\Public\Documents\Velox\NewProject.spp
```

GetSoftwarePath

Description

Returns the path for either the applications/data or project files.

Parameter Input

PathType

User (default)
Applications
Data

Parameter Output

SoftwarePath

Example

```
GetSoftwarePath User
```

GetStatus

Description

Returns the current software status. The server holds and maintains the software status. Status can be changed by using any of the following commands: BeginProbing, LoginDialog, AbortProbing, SetExternalMode, PauseProbing, UnloadWafer, ResumeProbing.

Parameter Output

N/U

Not used. (Always returns zero.)

Running Mode

Not used. (Always returns zero.)

Access Level

0 – Operator
1 – Engineer
2 – Admin
3 – Service

ControlMode

0 – Local
1 – Remote

LicenseDaysLeft

Example

```
GetStatus
0 0 0 0
```

IsAppRegistered

Description

Checks the server to see if the application `AppName` is registered with the server.

Parameter Input

AppName

Parameter Output

Result

0 – No

1 – Yes

Example

```
IsAppRegistered WaferMap
1
```

LoginDialog

Description

Displays the Login window and sends a New Access Level notification if the user enters a valid password. Four user levels are available: Operator, Engineer, Admin, and Service.

Parameter Input

LevelToOffer

Example

```
LoginDialog
```

OpenProject

Description

Opens the specified project.

Parameter Input

Filename

Example

```
OpenProject C:\Users\Public\Documents\Velox
```

OpenProjectDialog

Description

In the Open Project window, select the project file and click OK to open the project.

Parameter Input

Filename (optional)

Option

0 – Use the open project dialog (default).

1 – Open the specified project file (no dialog).

2 – Open the specified project file if it exists, otherwise use dialog.

Example

```
OpenProjectDialog
```

ReportSoftwareVersion

Description

Returns the full Velox software version.

Example

```
ReportSoftwareVersion
2.1.0.23
```

SaveProject

Description

Saves the current data to the project file.

Parameter Input

ProjectFile

Example

```
SaveProject C:\Project1.spp
```

SaveProjectAsDialog

Description

In the Save Project window, enter the project file name and click OK to save the project.

Example

```
SaveProjectAsDialog
```

SetAlignmentMode

Description

Sets the active alignment mode (either on axis or off axis).

Parameter Input

AlignmentMode

0 – Off Axis

1 – On Axis

Example

```
SetAlignmentMode 1
```

Notifications

Note

Never invoke notifications manually.

KernelConnectionStatus

Description

An example of Kernel Connection Status is "Windows Socket Error Number and Error Description." If StatusResult changes, and if StatusResult= Failed, retest result every 2 seconds until StatusResult =Succeeded.

Parameter Input

Status Type

S – SocketConnection status changes

Connected (C)

Disconnected (D) (default)

Temporary unusable (T)

P – Ping for Diagnostics

Succeeded (S)

Failed (F)

Example

```
KernelConnectionChange: S D (Disconnected: SocketError 10004 - WSAEINTR:
"Interrupted function call")
```

NewAccessLevel

Description

Alerts applications of new access level.

Parameter Input

Access level

1 – Operator

2 – Engineer (default)

3 – Admin

4 – Service

UserName

VeloxLocked

0 – Unlocked

1 – Locked

Example

```
NewAccessLevel 1
```

NewProjectFile

Description

Alerts applications if the project file was changed.

Parameter Input

FileName (optional)

If empty, project is untitled.

Example

```
NewProjectFile C:\Users\Public\Documents\Velox\NewProject.spp
```

PSProgressChanged

Description

Notification sent during wafer stepping process. It updates the progress information. 'Progress' is the relationship of dies tested to dies to be tested.

Parameter Output

Progress

Process progress in percent

0 if no process is pending

Example

```
PSProgressChanged 23.5
```

PSStateChanged

Description

Notification sent when VeloxPro changes its running state.

'Substatus' is required when 'Status' is ERROR, since 'Status' ERROR is set at the time of ERROR occurrence and not at sequence end. Within the ERROR status VeloxPro may still doing some cleanup tasks, which means it is actually BUSY or when the sequence has really finished the 'Substatus' is IDLE.

Parameter Output

Status

UNKNOWN – Process station is in unknown state when not initialized.

IDLE – Process station is in idle state when no process is running.

BUSY – Process station is in busy state when processes are running.

ERROR – Process station is in error state when an error occurs.

Not Initialized – The station is not initialized.

PAUSED – Process station is in paused state.

Substatus

UNKNOWN – Process station is in unknown state when not initialized.

IDLE – Process station is in idle state when no process is running.

BUSY – Process station is in busy state when processes are running.

ERROR – Process station is in error state when an error occurs.

Not Initialized – The station is not initialized.

PAUSED – Process station is in paused state.

Information

Contain descriptions of the current state; empty, by default.

Error messages are also displayed.

Example

```
PSStateChanged Error Busy "Profiling failed."
```

RegisterProberAppChange

Description

Alerts applications that an application is registered or unregistered on MsgServer.

Parameter Input

Application

Program name

Command Destination

Command section in command file

Register Status

0 – Unregistered

1 – Newly registered on MsgServer

Example

```
RegisterProberAppChange WaferMap WaferMap 1
```

SaveProjectFile

Description

Alerts applications to save the current project.

Parameter Input

FileName (optional)

If empty, saves under current name.

Example

```
SaveProjectFile C:\Users\Public\Documents\Velox\NewProject.spp
```

TTLTestDone

Description

Alerts applications that the TTL Test is ready.

Parameter Input

Error Code

TTL Test Error Code

Example

```
TTLTestDone 703
```

WMNewCurrentDie

Description

Alerts applications that WaferMap has a new position.

Parameter Input

X

A current die X coordinate

Y

A current die Y coordinate

XHome
The chuck X coordinate of the current position in microns from the home

YHome
The chuck Y coordinate of the current position in microns from the home

Site
A current site

LastSite
An index of the last site in the die

Example

```
WMNewCurrentDie 1 3 -5000 5030 2 4
```

WMSetupChange

Description

Alerts applications that a wafer's parameters have been changed.

Example

```
WMSetupChange
```

Chapter

8 Application Commands

The following command sets are included in this section:

- Optical Control Commands
- Scripting Console Commands
- Spectrum Commands
- TTL Interface Commands
- VeloxPro Commands
- Wafer Map Commands
- Profiling Commands

Optical Control Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands			
AZoomSetupDialog	MoveAZoomVelocity	OCLightVal	SelectAZoomLens
GetAZoomLens	OCGetZoomLevel	OCSetZoomLevel	SetAZoomLight
MoveAZoomFocus	OCLightOn	ReadAZoomFocus	StopAZoom

AZoomSetupDialog

Description

Opens the operator panel. In this window, you can change the settings for all defined lenses.

Example

```
AZoomSetupDialog
```

GetAZoomLens

Description

Returns the index of the currently-selected lens (1-4) preset button.

Parameter Output

Lens ID

Example

```
GetAZoomLens
1
```

MoveAZoomFocus

Description

Changes the focus magnitude.

Parameter Input

Focus

Percent magnitude. Default is 100.

Ref

Position reference.

R – Relative (default)

Z – Zero

Parameter Output

RetFocus

Percent magnitude response.

Example

```
MoveAZoomFocus 50 Z
50
```

MoveAZoomVelocity

Description

Moves the A-Zoom using the set focus speed.

Parameter Input

Direction (required)

Direction '+' or '-'.

Velocity (optional)

Percent focus speed. Default is 100.

Parameter Output

RetFocus

Response Percent magnitude.

Example

```
MoveAZoomVelocity + 67
67
```

OCGetZoomLevel

Description

Gets the logical zoom factor for the given motor.

Parameter Input

Motor

0 – Off

1 – On (default)

Parameter Output

Zoom

Example

```
OCGetZoomLevel 1
```

OCLightOn

Description

Switches the light On or Off in a given channel. (If it has segments, then all segments will be switched On or Off.)

Parameter Input

On

0 – Off (default)

1 – On

Chn

Channel number (hardware dependent). Default is 1.

Example

```
OCLightOn 1 1
```

OCLightVal

Description

Sets the brightness of the lamp to the given value as a percentage. Sets the brightness of all segments (if there is more than one) from first light to value 32.

Parameter Input

Val

Logical brightness (0–100). Default is 0.

Chn

Channel number (hardware dependent). Default is 1.

Segment

Segment number (hardware dependent). Default is 1.

Example

```
OCLightVal 128 1 4
```

OCSetZoomLevel

Description

Sets the zoom level to the given percentage.

Parameter Input

Zoom

Logical zoom (0–100). Default is 0.

Motor

Motor ID. Default is 1.

Example

```
OCSetZoomLevel 50 1
```

ReadAZoomFocus

Description

Returns the focus magnitude (percent).

Parameter Output

Focus

Example

```
ReadAZoomFocus
99
```

SelectAZoomLens

Description

Selects the Lens Preset (1-4) specified by the given index number.

Parameter Input

Lens ID

Default is 1

Example

```
SelectAZoomLens 1
```

SetAZoomLight

Description

Switches all lights on or off. If no parameter is specified, toggles between on an off.

Parameter Input

Light status

0 – Switches the light off (default)

1 – Switches the light on

Example

```
SetAZoomLight 1
```

StopAZoom

Description

Stops the A-Zoom movements.

Example

```
StopAZoom
```

Scripting Console Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands			
CloseCommunicator	DoScript	GetRunStatus	StartScript

CloseCommunicator

Description

Command closes the Scripting Console program. This is used during the Project File handling.

Example

```
CloseCommunicator
```

DoScript

Description

Executes the script and returns the response afterwards.

Parameter Input

Script name

Example

```
DoScript "myscript.scr"
```

GetRunStatus

Description

Returns information about Scripting Console status.

Parameter Input

Script name

Currently executed script

Parameter Output

Running

Is Scripting Console running the script

0 – No

1 – Yes

LastError

Last error returned by the script

Script name

Example

```
GetRunStatus MyScript
1 0 myscript.scr
```

StartScript

Description

Starts the script and returns the response immediately.

Parameter Input

Script name

Example

```
StartScript myscript.scr
```

Spectrum Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Camera Commands
General Commands
ReAlign Commands
Tool Commands

Camera Commands

Commands		
GetCameraLight	SetCameraLight	SetCameraView
GetCameraView	SetCameraQuiet	

GetCameraLight

Description

Returns light properties of chosen camera.

Parameter Input

Tool name (case sensitive) or Camera mount position (optional)

Select a camera by its mount position (Scope, Chuck, Platen, ContactView) or use tool camera by giving tool name, such as AlignWafer.

Parameter Output

Camera mount position

Mount position of the camera where the light settings are from

State

0 – Lights off

1 – Lights on

ShutterTime

Returns shutter time (1.0 – 20.0) in ms

Gain

Returns gain (0.0 –100.0)

Example

```
GetCameraLight Scope
Scope 0 12.8 5.2
```

GetCameraView

Description

Returns the present camera view state of a desired tool or active view.

Parameter Input

Tool name (case sensitive) or Camera mount position (optional)

Select a camera by its mount position (Scope, Chuck, Platen, ContactView) or use tool camera by giving tool name, such as AlignWafer.

Parameter Output

Camera mount position

Camera mount position of present view (Scope, Chuck, Platen, ContactView)

Zoom (Zoom factor of present view)

-2 – 50%
-1 – 100%
0 – no change
1 – 100%
2 – 200%

LiveVideo

0 – Off
1 – On

WindowState

0 – Hidden
1 – Normal
2 – Maximized

Example

```
GetCameraView
Platen -1 1 2
```

SetCameraLight

Description

Switches the light of the chosen camera on or off. When lights are switched on, shutter time and gain are set as well.

Parameter Input

Tool name (case sensitive) or Camera mount position

Select a camera by its mount position (Scope, Chuck, Platen, ContactView) or use tool camera by giving the tool name, such as AlignWafer

State (optional)

0 – lights off (default)
1 – lights on

ShutterTime (optional)

Set Shutter time (1.0 – 20.0) in ms (default: -1)

Gain (optional)

Set Gain (0.0 –100.0) (default: -1)

Example

```
SetCameraLight AlignWafer 1 7 5
SetCameraLight Platen 1 7.0 5.0
```

SetCameraQuiet

Description

Activates/deactivates the camera quiet mode. (Applies only to ATM300A, SPS300, and SUMMIT200 stations.) The camera quiet mode deactivates the chuck, platen and contact view camera and triggers a digital output which connects/disconnects the firewire connection to the PC.

Parameter Input

Activate camera quiet mode

If true, activates the camera quiet mode. (default)

If false, camera quiet mode is deactivated.

Example

```
SetCameraQuiet 1
```

SetCameraView

Description

Switches the desired video window of Spectrum as foreground and active view. The camera view can be determined by the camera mount position or by a tool such as AlignWafer. The Zoom parameter defines the display's zoom factor. The LiveVideo parameter lets you toggle the live video. The WindowState lets you maximize a window.

Parameter Input

Camera mount position or tool name (case sensitive)

Scope, Chuck, Platen, ContactView camera to be set as: active camera, or tool, such as, AlignWafer

Zoom (optional)

-1 – FitImage
0 – No change (default)
1 – 100%
2 – 200%

LiveVideo (optional)

0 – Off
1 – On
2 – Keep current (default)

WindowState (optional)

0 – Hidden (default)
2 – Maximized

Example

```
SetCameraView AlignWafer -1 2 or SetCameraView Scope -1 2
```

General Commands

Commands	
EnableOverlay	SetAutomationActive
GetAutomationActive	SetConstantContactMode
GetAutomationTemperatureStatus	ShowWizard
GetConstantContactModeStatus	StartAutomationTemperature
ResetAutomation	StopAutomationTemperature

EnableOverlay

Description

Allows activating the overlay of the needles for the top camera.

Parameter Input

Overlay index

0 – Deactivate overlay (default)
1 – Overlay for ProbeMark 1
2 – Overlay for ProbeMark 2

Example

```
EnableOverlay 0
```

GetAutomationActive

Description

Returns the current state of Automation mode. If automation is enabled (AutoXY, AutoZ or VueTrack), die stepping moves are compensated by Wafer Map.

Parameter Output

Status

0 – If disabled, Automation is not compensating for die stepping.
1 – If enabled, Automation is compensating for die stepping.

Example

```
GetAutomationActive
0
```

GetAutomationTemperatureStatus

Description

Returns the status of the current temperature ramping using automation process.

Parameter Output

Status ID

0 – Complete
1 – Busy
2 – Error

Status

Complete – The temperature transition operation finished successfully.

Busy – The operation is still not finished.

Error – On error, will return a description of the error that occurred

Example

```
GetAutomationTemperatureStatus
```

GetConstantContactModeStatus

Description

Returns the status of the current ConstantContact automation process.

Parameter Output

Status ID

0 – Complete

1 – Busy

2 – Error

Status

Complete – The temperature transition operation finished successfully.

Busy – The operation is still not finished.

Error – On error, will return a description of the error that occurred

Example

```
GetConstantContactModeStatus
```

ResetAutomation

Description

Reset automation data back to trained values

Example

```
ResetAutomation
```

SetAutomationActive

Description

This command can be used to activate or deactivate Automation (AutoXY, AutoZ or VueTrack). If Automation is not trained, an error will be returned when activating.

Parameter Input

Status

0 – Disable automation (default)

1 – Enable automation

Example

```
SetAutomationActive 1
```

SetConstantContactMode

Description

Starts Constant Contact mode using Automation. Constant Contact mode will adjust the wafer X, Y, and Z positions to keep the trained target at a consistent location relative to the scope during thermal transitions. Constant Contact mode can be enabled when the chuck is at contact or at separation.

Parameter Input

Mode

0 – Disable constant contact mode (default)

1 – Enable constant contact mode

Example

```
SetConstantContactMode 1
```

ShowWizard

Description

Starts the wizard of a given tool on the display defined in the tool settings. Additionally, it can start a wizard for training single models with the syntax, for example, ShowWizard FindFeature ProjectData/Features/Model1.

For the tools MeasureOnScreen, Calibrate, and CameraOrigin, you can specify a mount position, for example, ShowWizard Calibrate Platen.

Parameter Input

Tool name (case sensitive)

Name of the tool from which the wizard will be started. Alternatively, the name of the model, including its data path, can be specified (see example).

Camera mount position (optional)

Camera mount position on which the wizard will run. Supported for MeasureOnScreen, CameraOrigin, and Calibrate.

AskExecute

Default is 0

Parameter Output

Wizard result

0 – Wizard finished successfully

1 – Wizard canceled

Examples

Display the ProbeCardTool wizard

```
ShowWizard ProbeCardTool
0
```

Display the model training wizard for FindFeature (first feature)

```
ShowWizard FindFeature/ProjectData/Features/Model1
0
```

Display the model training wizard for the first alignment mark

```
ShowWizard AlignWafer/ProjectData/Mark1/Model1
0
```

StartAutomationTemperature

Description

Starts temperature ramping using Automation + VueTrack. Automation readjusts the probes and wafer while ramping to the desired target temperature. Command returns immediately.

Parameter Input

TargetTemperature (required)

Enter the desired temperature.

ThetaAlignOnFinish

0 – False (default)
1 – True (If true, it performs a theta alignment operation at the completion of the transition.)

ContactOnFinish

Defines whether or not the chuck will move back up into contact at the completion of the transition.

0 – False (default)
1 – True (If true, the chuck will move to contact.)

AlignOnFinish

Defines if final adjustments are made after temperature transition.

0 – False (does temperature transition corrections, but does not make calculations for final touchdown)
1 – True (default)

Example

```
StartAutomationTemperature 150
```

StopAutomationTemperature

Description

Stops temperature ramping when in the middle of an Automation Temperature transition operation.

Example

```
StopAutomationTemperature
```

SwitchOffset

Description

Switches the offset compensation. Applicable for MicroAlign stations only.

Parameter Input

Offset Position

0 – Disable offset (default)
1 – Probe Mark 1
2 – Probe Mark 2
3 – Die Center

Example

```
SwitchOffset 1
```

ReAlign Commands

Commands			
GetReAlignStatus	NextWafer	StartReAlign	StartReAlignTemperature
GetReAlignTemperatureStatus	ReAlign	StopReAlign	StopReAlignTemperature

GetReAlignStatus

Description

Returns a status of the ReAlign procedure. If the return value is true, ReAlign is running.

Parameter Output

Running

0 – False, ReAlign is not running

1 – True, ReAlign is currently running

Example

```
GetReAlignStatus
0
```

GetReAlignTemperatureStatus

Description

Returns a status of the current temperature ramping using ReAlign process.

Parameter Output

Status

0 – Complete

1 – Busy

2 – Error

Status Message

Will return an error message if there is an error.

Example

```
GetReAlignTemperatureStatus
0
```

NextWafer

Description

Starts the NextWafer routine when the wizard is finished or cancelled.

Parameter Output

Cancelled

0 – False.

1 – True, the wizard is finished or cancelled by the user.

Example

```
NextWafer
0
```

ReAlign

Description

Performs an automatic realignment of the wafer. This includes measuring the needle drift XYZ, aligning the wafer, measuring chuck expansion, measuring chuck drift XYZ, and optionally, a Z-Profile.

Alternatively the tool performs a ProbeToDie alignment to correct the needle position for the current die. Command returns once ReAlign is finished, or aborted if there is an error.

Parameter Input

Attempts on error

1-10, default is 1

Mode

H – Perform Full ReAlign including theta alignment (default)

C – Perform ProbeToDie Align for current die

AlignProbeCard

Update the probe card position.

0 – False

1 – True

2 – Use tool settings (default)

CorrectZ

Calculate a new contact height (focus on wafer and/or needle)

0 – False

1 – True

2 – Use tool settings (default)

Parameter Output

X Offset wafer

Measured chuck offset of the die position in x

Y Offset wafer

Measured chuck offset of the die position in y

Z Offset wafer

Measured chuck offset of the die position in z

X Offset probe card

Drift of the probe card in x

Y Offset probe card

Drift of the probe card in y

Z Offset probe card

Drift of the probe card in z

Examples

Performs a realignment for the wafer.

```
ReAlign 2
-1 -1 -1 -1 -1 -1
```

Performs a ProbeToDie alignment at the current die with correction of probe card and wafer position in xy.

```
ReAlign 2 C 1 0
0.1 -0.2 0 0.3 -0.25 0
```

StartReAlign

Description

Starts an automatic realignment of the wafer. This includes measuring the needle drift XYZ, aligning the wafer, measuring chuck expansion, measuring chuck drift XYZ, and optionally, a Z-Profile.

Alternatively the tool performs a ProbeToDie alignment to correct the needle position for the current die. Command returns immediately.

Parameter Input

Attempts on error

1-10. Default is 1

Mode

*H – Perform full ReAlign including theta alignment (d*efault)

C – Perform ProbeToDie align for current die

AlignProbeCard

Update the probe card position.

0 – False

1 – True

2 – Use tool settings (default)

CorrectZ

Calculate a new contact height (focus on wafer and/or needle)

0 – False

1 – True

2 – Use tool settings (default)

Example

```
StartReAlign 1 H
```

StartReAlignTemperature

Description

Starts temperature ramping using ReAlign. ReAlign will readjust the probes and wafer during ramping to the new target temperature. Command returns immediately. Command returns immediately.

Parameter Input

TargetTemperature

The target temperature for heating or cooling the chuck. Default is 0.

Align Theta on Finish

Defines if a theta alignment is executed when finished with thermal ramping sequence.

0 – False (default)

1 – True

Move to Contact on Finsh

Defines if the chuck moves into contact at completion of thermal ramping sequence.

0 – False (default)

1 – True

AlignOnFinish

Defines if final adjustments are made after temperature transition. If false, then it does temperature transition corrections only, but does not make calculations for final touchdown.

0 – False

1 – True (default)

Example

```
StartReAlignTemperature 150
```

StopReAlign

Description

Stops the ReAlign procedure.

Example

```
StopReAlign
```

StopReAlignTemperature

Description

Stops temperature ramping using ReAlign.

Example

```
StopReAlignTemperature
```

Tool Commands

Commands			
AlignAux	DetectWaferHeight	LocateHomeDie	ReadBarCode
AlignAuxOffAxis	FindFeature	PreMapWafer	ReadOcrString
AlignChip	FindFocus	ProbeCardOCS	ShapeTracker
AlignWafer	FindWaferCenter	ProbeToPadAlign	SnapImage
AutoAlign	GetPreMappedDieInfo	Read2DMatrixCode	SynchronizeCamera
AutomationSearchCurrentDie	GetWaferCenter		VueTrackAlign

AlignAux

Description

Performs an automated aux site alignment in XYZ. Tool can correct reference position and contact height for the given aux site.

Parameter Input

AuxSite

Aux site to be automatically aligned.

Example

```
AlignAux 1
```

AlignAuxOffAxis

Description

Performs an automated aux site alignment in XYZ. Tool can correct reference position and contact height for the given aux site.

Parameter Input

AuxSiteID

Aux site is automatically aligned.

Example

```
AlignAuxOffAxis 1
```

AlignChip

Description

Performs a single or two point alignment and moves the current chip to the trained aligned position in Theta (in degrees) and X, Y (in microns). The current chip is assumed to be in the region of interest when the command is called.

Parameter Output

Theta Offset

Difference in theta from before AlignChip command is sent to after AlignChip execution.

X Offset

Difference in X from before AlignChip command is sent to after AlignChip execution.

Y Offset

Difference in Y from before AlignChip command is sent to after AlignChip execution.

Example

```
AlignChip
-0.025 67.0 -5.0
```

AlignWafer

Description

Performs a two-point alignment and moves chuck to the trained align position. XY correction is made on *Track Position =True*. Returns the theta corrected value in degrees.

Parameter Input

Track Position

Track trained chuck position (overwrite) after execution to get right start position after wafer was aligned.

Do not use for fully automated systems.

Parameter Output

Theta Offset

Difference in theta from before the AlignWafer call to after AlignWafer execution.

Example

```
AlignWafer
0.567
```

AutoAlign

Description

Executes the AutoAlign tool to do a theta alignment of the wafer with optional index calculation or thermal expansion measurement. The command will move to its trained chuck/scope position before starting the alignment if the option MoveToTrainedXYPosition and/or MoveToTrainedZPosition is true, or if the command is setting the home position.

Parameter Input

Value To Set

0 – Do Alignment (default)
1 – Do Alignment and Set Index
2 – Do Alignment and Set Thermo Scale Value

Skip Setting Home

0 – Set Home Position (default)
1 – Don't set Home

Example

```
AutoAlign 2
1
```

AutomationSearchCurrentDie

Description

Performs an Automation search and position correction using the current chuck position. Response is X,Y error offset of the current die.

Parameter Output

X Offset

Y Offset

Example

```
AutomationSearchCurrentDie
```

DetectWaferHeight

Description

Synchronizes chuck and top camera in X, Y and Z. Then the wafer surface is focused on the trained chuck position or the manual adjusted reference position. This tool uses the FindFocus tool with its own Range.

Parameter Input

Set chuck start position

0 – Use trained chuck position (default)
1 – Choose new start position
2 – Use given position

Synchronize

0 – Do not synchronize (default)
1 – Synchronize

Chuck position X

X Position where Wafer is to be focused (default: -1)

Chuck position Y

Y Position where Wafer is to be focused (default: -1)

Parameter Output

Chuck position X

Chuck X position where wafer was focused

Chuck position Y

Chuck Y position where wafer was focused

WaferHeight
Focus height

SynchWaferGap
Gap between Synchronize Z position and focus height when Synchronize = true

ZOffset
Offset to old chuck Z position

Stage
Stage used to perform focus search:
S – Microscope
C – Chuck

Example

```
DetectWaferHeight 1 1
190326 163812 17054 1382 -24.6 C
```

FindFeature

Description

Stores user-defined models. Up to 40 different models can be trained. After training, these models can be searched on screen either by direct user interaction or from external applications with a remote command.

The remote command returns the distance from the center of the field of view (FOV) to the match center, angle, and score value for each match found in the field of view.

Parameter Input

FeatureNumber
Default is 1

Return distance from model origin
1 – Returns distance from match to training location of model.
0 – Returns distance from match to center of field of view. Default is 0.

UseSingleImageAcquisition
1 – Single Image Acquisition (true) (default)
0 – Live image (false)

Parameter Output

X Position
Lateral distance in microns from the match to the center of the field of view.

Y Position
Vertical distance in microns from the match to the center of the field of view.

Angle
Angle of the match in degrees.

Score Value
Match score. Zero for worst possible score, and one for the best possible score.

Example

```
FindFeature 1 0 0
58.78,-73.35,-0.0029978,0.997;62.84,172.38,0.0873014,0.786
```

FindFocus

Description

Determines the Z axis position where an object in the region of interest is in focus. The output parameter is the Z axis value in microns from zero of the new focus height. The second output parameter is the stage used to perform focus search.

Parameter Input

StepCount

Optional step count (default: -1)

Range

Optional scan range (default: -1)

Parameter Output

Z position

Z value when object is in focus

Stage

Stage used to perform focus search:

S – Microscope

C – Chuck

Example

```
FindFocus
122.0 C
```

FindWaferCenter

Description

Finds the center of the wafer using edge detection.

Parameter Input

NoManualRecovery

0 – User will be prompted to manually locate the edge if we fail to find an edge.

1 – No manual edge selection will occur if we fail to find an edge.

Parameter Output

ChuckX

X position where Wafer Center is found.

ChuckY

Y position where Wafer Center is found.

Example

```
FindWaferCenter
```

GetWaferCenter

Description

Returns the chuck location (zero based) of the wafer center. It returns 0 0 if FindWaferCenter has not been executed for the current wafer.

Parameter Output

ChuckX

X position where Wafer Center is found.

ChuckY

Y position where Wafer Center is found.

Example

```
GetWaferCenter
```

LocateHomeDie

Description

Finds the center of the wafer using edge detection and sets the home position. This updates the home position for the home die as defined by wafer map. The default Die Map specifies the upper left corner of the home die. The wafer map must be an accurate representation of the physical wafer layout for proper operation.

Parameter Input

NoManualRecovery

0 – User will be prompted to manually locate the edge if we fail to find an edge.

1 – No manual edge selection will occur if we fail to find an edge.

Parameter Output

ChuckX

X position where Wafer Home is found.

ChuckY

Y position where Wafer Home is found.

Example

```
LocateHomeDie
```

PreMapWafer

Description

PreMapWafer finds actual singulated die locations and updates positions used by WaferMap.

Parameter Output

NumberOfDies

Example

```
PreMapWafer
0
```

ProbeCardOCS

Description

Determines the chuck XYZ axis position where the needles of the probe card are in focus. The output parameter is the Z axis value in microns from zero of the new focus height.

Parameter Input

UpdateZ

Do a focus search on the needles and update the needle height.

MeasureBothGroups

Do a focus search on both groups of needles or only one. Default is 1.

Parameter Output

Z axis Position

Chuck height (µm from Zero) where the needles are in chuck camera focus.

Example

```
ProbeCardOCS
10584.0
```

ProbeToPadAlign

Description

Sets a new Home position. The command is used during ReAlign to search the trained home reference model and to calculate the new xy home position.

Parameter Input

Position (optional)

H – ProbeToPadAlign at home die

C – ProbeToPadAlign at current die

Parameter Output

Offset X

X offset from the expected wafer position to the identified wafer position

Offset Y

Y offset from the expected wafer position to the identified wafer position

Example

```
ProbeToPadAlign
6.5 -5.3
```

Read2DMatrixCode

Description

Reads the wafers matrix code and decodes it to a string. Requires IDTools license.

Parameter Output

2DmatrixCodeString

Example

```
Read2DMatrixCode
54SFK52
```

ReadBarCode

Description

Reads the wafer's barcode and decodes it to a string. If empty string is returned, then no string or code was successfully read. Requires IDTools license.

Parameter Output

BarCodeString

Example

```
ReadBarCode
123AAC7
```

ReadOcrString

Description

Reads the wafer ID string from the wafer's surface. Returns the read ID as string. If empty string is returned, then no string or code was successfully read. Requires IDTools license.

Parameter Output

OcrString

Example

```
ReadOcrString
205-167985-18F
```

ShapeTracker

Description

Tracks the shape of a substrate (such as a broken wafer) to determine the contour of it. The data will be remembered as xy coordinates of contour points referenced to the home position. These points are written to a file after tracking the shape.

Parameter Input

SetHome (optional)

0 – System will *not* set the current chuck position to Home before tracking.

1 – System will set the current chuck position to Home before tracking.

AutoEdgeFind (optional)

0 – The edge must be found in the current field of view (default).

1 – The edge of the substrate is detected automatically by indexing west of the current position.

FileName (optional)

If empty setup parameter is used, defines the full path of the result file where the contour points are stored.

Example

```
ShapeTracker 1 0 "c:\shape.log"
```

SnapImage

Description

Saves the currently displayed image to the specified file. The image is stored in the requested file format (bmp, jpg or png). It will save the raw camera image and an image with the overlays that are currently visible on the camera view.

Using a parameter, one can decide to save either the raw image, the overlay image, or both images.

By specifying 'ALL' as the mount position, the captured screenshot will consist of the currently-selected camera layout without providing the raw image.

Parameter Input

Mount position

Mount position of camera from which the image will be taken: Scope (default), eVue1, eVue2, eVue3, Chuck, Platen, ContactView, or All. (If you use 'Scope' on a system with eVue, the image will be from eVue2).

If set to 'All' the currently visible camera layout will be captured in the screenshot.

FullPath

Path where the image will be stored.

SnapshotMode

0 – Raw Image from camera;

1 – Screenshot of camera window including overlays;

2 – Both images (default)

Example

```
SnapImage Scope C:\Image1.bmp
```

SynchronizeCamera

Description

Synchronizes the Platen camera with the calibration mark. The mark is mounted on the chuck camera. The chuck must be moved to get the mark in view of the specified camera.

Parameter Input

Camera mount position

Camera (scope, chuck, platen) to be synchronized. Default is scope.

Synchronize XY

Synchronize *Camera* in xy (default is 0)

Synchronize Z

Synchronize *Camera* in z (default is 0)

Parameter Output

X Position

New camera X coordinate (technolog. from zero)

Y Position

New camera Y coordinate (technolog. from zero)

Z Position

New camera Z coordinate (technolog. from zero)

Example

```
SynchronizeCamera 1 1 0
305000 280050 268790
```

VueTrackAlign

Description

Sets alignment for the next wafer when using VueTrack.

Parameter Input

FullVueTrackAlign (optional)

0 – Perform only chuck theta align

1 – Perform chuck theta align, AutoXY, AutoZ, VueTrack (default).

Example

```
VueTrackAlign 1
```

TTL Interface Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands		
CancelTTLTest	GetTTLLines	SetTTLLine
DoTTLTest	GetTTLStatus	StartTTLTest

CancelTTLTest

Description

Stops the wafer test.

Example

```
CancelTTLTest
```

DoTTLTest

Description

Starts execution of one test. It returns a bin number or multiple bin numbers when clusters are enabled in the WaferMap (one number for each die in a cluster).

Parameter Output

Bit number

Example

```
DoTTLTest
7
```

GetTTLLines

Description

Returns the I/O lines as bit array.

Parameter Output

LineA

LineB

LineC

Example

```
GetTTLLines
1 2 3
```

GetTTLStatus

Description

Returns the error 914 if the test is in progress.

Parameter Output

Zero if the last test is finished

Error code if it was canceled or an error occurred

Example

```
GetTTLStatus
0
```

SetTTLLine

Description

Sets the selected line of PA port.

Parameter Input

Line

0…7

Value

0 or 1

Example

```
SetTTLLine 1 1
```

StartTTLTest

Description

Starts the test of a wafer. Depending on the cluster definitions queried from the WaferMap, all clusters or all dies will be tested.

Example

```
StartTTLTest
```

VeloxPro Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands

AbortJob	GetJobParams	ProceedProbing	StartWaferJob
ConfirmRecipe	GetProbingStatus	ProcessWafer	TransportWafer
DockCassette	JobStatus	QueryCassetteID	UnloadWafer
GetCassetteStatus	LoadWafer	QueryWaferID	UpdateCassetteStatus
GetJobList	ProceedJob	QueryWaferInfo	UpdateWaferID

AbortJob

Description

Aborts the job given by the job ID. The status of the job will be set to “Aborted.” The “Unload” parameter defines whether the wafer remains on the chuck or not.

Parameter Input

Job ID

The ID of the job to abort.

Unload Wafer

True – Unload the wafer after aborting

False – The wafer stays on the chuck

Example

```
AbortJob 187 1
```

ConfirmRecipe

Description

Queries whether the given project/flow name is a valid project/flow at the probe station.

Using file and folder names without white spaces is recommended. For folders and files with white spaces, use quotation marks for the path.

Parameter Input

Recipe Name

Full path of the recipe file

Parameter Output

Valid

True if the recipe is valid at the probe station

Error Message

Message that contains the reason why the recipe is not valid

Example

```
ConfirmRecipe C:\Test.spp
```

Example for using white spaces:

```
ConfirmRecipe "C:\\Users\\Public\\Documents\\Velox\\Projects\\White Spaces.spp"
```

DockCassette

Description

Docks or undocks a cassette which is placed on a LoadPort (MHU300, MHU301). It will return an error if no cassette is placed: the load port is not ready/available or the docking failed. The command will do the docking/undocking and return when this is done.

Parameter Input

Loadport ID

ID of the load port where the docking should take place

Dock/Undock

Dock or undock the cassette

Example

```
Dock the cassette on Loadport 1
DockCassette 1 1
```

GetCassetteStatus

Description

Provides all of the available wafer information. It returns a set of data for each wafer in either cassette. The data contains the cassette and the slot number where the wafer is located, the status of the wafer, and identification information.

Parameter Output

Cassette Status

String contains all relevant information of the cassette. The string is separated by semicolon.

Cassette – 1 or 2
Slot – 1-25
SlotStatus – Empty, Present, Testing, Error
WaferIDStatus – Pending, Ok, Error
WaferID – String

Cassette Slot SlotStatus WaferIDStatus WaferID; Cassette Slot SlotStatus WaferIDStatus;…

Example

```
GetCassetteStatus
1 8 Present Ok XY1234; 1 9 Error Ok xy1235; 1 10 Testing Pending
```

GetJobList

Description

Returns a string for the relevant JobIDs.

Parameter Input

JobType

The type of job to query (default is Probing). Use 'All' to see all job types.

Parameter Output

JobIDs

Contains all relevant JobIDs of the job queue

Example

```
GetJobList
1 Done;2 Aborted
```

GetJobParams

Description

Get job information for a specific job ID including the flow file and the list of wafers.

Parameter Input

Job ID

The ID of the job to query

Parameter Output

Recipe Name

Name of the recipe for this job

Wafers

List of all wafers that are included in this job. Wafer IDs are separated by a single space.

Example

```
ConfirmRecipe C:\Test.spp
```

GetProbingStatus

Description

Retrieves the current status of the prober. The tester can take control if probing status is TesterInControl or AtFirstDie.

Parameter Output

Probing Status

U = Unknown – Prober state is unknown
B = Busy – Prober is busy
I = Idle – Prober is idle
E = Error – Prober has encountered an error
A = AtFirstDie – Prober waits for the command ProceedProbing to proceed with the recipe
T = TesterInControl – Allows tester to use the prober for stepping and testing

Example

```
GetProbingStatus
TesterInControl
```

JobStatus

Description

Retrieves the status of a specific job by its ID. If the status is ErrorWaiting, an error string is returned.

Parameter Input

Job ID

The ID of the job to query

Parameter Output

Job Status

Pending
Running
Paused
Aborting
Aborted
Done
Invalid
ErrorWaiting

Status Info

Only if Job Status is ‘ErrorWaiting’

Info = Quoted Error String with following format:

“<ProcessModule><space><ErrorNumber><space> <Description><semicolon><SubProcessModule><space> <SubErrorNumber><space><SubErrorDescription>”

The String contains 2 data sets.They are separated by semicolon. The separator inside the data set is a space.

- The first data set is always available and describes the error on the LoaderModule Level.
- The second data set is additional available, if the error has been generated by a module (VeloxPro).

ProcessModule (and SubProcessModule) can be:

- Cassette1
- Cassette2,
- ProberCurrent (Prober, which VeloxPro controls)
- IDReader
- PreAligner
- QuickAccessPort
- Robot
- Host

Example

JobStatus 123

```
ErrorWaiting "Host 4406 ProberError: Processing VeloxPro recipe caused an error;
ProberCurrent 17 Feature unavailable with current hardware configuration"
```

Host means that the error was generated by the host itself. The host is the LoaderModule.exe that keeps the modules.

The modules are the concrete HW Modules: Roboter, Prober1, 2, Cassette1,2, PreAligner,IDReader

JobStatus 123

```
ErrorWaiting "Host 4406 ProberError: Processing VeloxPro recipe caused an
error;ProberCurrent 853 Focus not found”
```

The execution of the recipe was canceled because Spectrum has not found the focus.

JobStatus 123

```
ErrorWaiting "ProberCurrent 4403 Electronics not responding;ProberCurrent 501
Command Response Timeout – MoveChuckLoad”
```

ProberCurrent is displayed because the error was generated by the module itself (CMD.Module.CMDProber.dll).

LoadWafer

Description

Load a wafer from a given load port (CM300, SUMMIT200, and BlueRay/APS200Tesla) onto the prober. If no slot ID is defined, the first available wafer with the lowest slot ID will be loaded. If no load port is defined, it will try to use the first load port. If there is no suitable wafer, it will try the second load port.

Returns an error if the loading process fails or no suitable wafer is found. To define an alignment angle and still use the "autoselect" behavior, use -1 for load port and slot.

Parameter Input

LoadportID

Default is -1

SlotID

Default is -1

AlignmentAngle
Default is 0

Example

```
LoadWafer 1 1
```

ProceedJob

Description

Continues a job that is waiting on an error (Job Status ErrorWaiting). The mode determines how the job continues.

Parameter Input

Corrective Action
AbortJob
AbortJobUnloadWafer
RemoveWafer
RetryAction
SkipError

Example

```
ProceedJob SkipError
```

ProceedProbing

Description

Tells the prober to proceed the recipe when current status is TesterInControl or AtFirstDie.

Example

```
ProceedProbing
```

ProcessWafer

Description

Process a wafer on a given location. The action depends on the current station configuration:

- IDReader: read wafer ID
- PreAligner: align wafer (alignment angle mandatory)
- Prober: Perform current recipe

Parameter Input

Module
Default is Cassette1

ProcessParam
Default is 0

Example

```
ProcessWafer PreAligner 90.0
```

QueryCassetteID

Description

Returns the ID of a cassette that is placed on the load port. The loadport and the cassette must support RFID reading (additional hardware needed).

Parameter Input

CassetteNumber

1 – LoadPort1 (default)

2 – LoadPort2

Parameter Output

CassetteID

Example

```
QueryCassetteID 1
CassetteABC
```

QueryWaferID

Description

Queries the wafer ID of a given module and slot.

Parameter Input

Module

Name of the module which is to be queried. For instance, Cassette1, Cassette2, Prober1, Prober2, IDReader, PreAligner, QuickAccessPort, Robot

Slot

Slot number of the wafer

Parameter Output

Wafer ID

ID of the wafer at the specified place

Example

```
QueryWaferID Cassette2 13
XY12345
```

QueryWaferInfo

Description

Returns information about the wafer that is currently on the probe station. It will return an error when there is no wafer currently on the chuck.

Parameter Output

Wafer size

Size of the wafer on the probe station

Wafer angle

Angle of the wafer on the probe station. This is the angle that the prealigner used for aligning the wafer before it was loaded on the chuck.

Wafer ID

ID of the wafer on the probe station

Lot ID

Lot ID of the wafer on the probe station

Product ID

Product ID of the wafer on the probe station

Example

```
QueryWaferInfo
300 0.0 Slot_101 Lot01 Product01
```

StartWaferJob

Description

Creates and starts a new job by specifying the flow/spp file and the wafers to be included in the job.

Parameter Input

Recipe Name

Name of the probe station project or flow for this job.

Wafers

List of all wafers that should be included in this job. Wafer IDs are separated by a single space. Alternatively, the wafers can be selected by their position in the cassette. For instance, '5;2' must be issued for wafer slot 5 in cassette 2.

Parameter Output

Job ID

The ID of the newly created job

Examples

```
StartWaferJob C:\Users\Public\Documents\Velox\Test.flow XY1234 XY1235 XY1236
23
```

```
StartWaferJob C:\Users\Public\Documents\Velox\Test.flow 1;1 2;1
24
```

TransportWafer

Description

Transport a wafer from one location to another. This command behaves like the wafer-transport inside the LoaderModule. The slot must always be set. Available stations are: Cassette1, Cassette2, Prober, IDReader, PreAligner, QuickAccessPort.

Parameter Input

SourceLocation

Loadport or Chuck. Default is Cassette1

SourceSlot

QuickAccessPort or Loadport. Default is 0

DestinationLocation

Loadport or Chuck. Default is Cassette1

DestinationSlot

Loadport or QuickAccessPort. Default is 0

Example

```
TransportWafer Cassette1 1 PreAligner 1
```

UnloadWafer

Description

Unloads a wafer from the chuck to its origin. If no origin is known, the first free slot of a loadport will be chosen. A wafer must be on the chuck.

Example

```
UnloadWafer
```

UpdateCassetteStatus

Description

Updates the cassette status with a simple scan.

Parameter Input

Cassette

Enter the cassette number for which to retrieve status. Default is 0.

Parameter Output

CassetteStatus

String contains all relevant information of the cassette

Example

```
UpdateCassetteStatus 1
1 8 Present Ok XY1234; 1 9 Error Ok xy1235; 1 10 Testing Pending
```

UpdateWaferID

Description

Overrides the wafer ID of the given module and slot.

If the wafer ID is empty and the module is a load port, the ID will be read by the IDReader, if possible. This process behaves as follows:

- The current wafer ID is reset.
- If the read wafer process is delayed (for example, a wafer is already on the IDReader) the process will be aborted and an error will be returned.
- If the ID reading fails, the process behaves like a normal inventory triggered by the UI:
  - Skip (returns success)
  - Abort (returns error)
  - Ask user

For remote ID reading, use either Skip or Abort.

If the module is a Loadport and the slot is set to -1, a regular inventory will be triggered. This is similar to an inventory triggered by the UI – the ID won't be reset and already-scanned wafers won't be reset. Setting an ID explicitly with slot -1 is an error:

Okay—read ID of wafer in slot 3 even if it is already set: UpdateWaferID Cassette1 3

Okay—set id of wafer in slot 3 to XYZ123: UpdateWaferID Cassette1 3 XYZ123

Okay—start full inventory: UpdateWaferID Cassette1 -1

Error—UpdateWaferID Cassette1 -1 XYZ123

Parameter Input

Module

Name of the module which is to be queried. For instance,

Cassette1, Cassette2, Prober1, Prober2, IDReader, PreAligner, QuickAccessPort, Robot

Slot

Slot number of the wafer

Wafer ID

ID of the wafer at the specified place

Example

```
UpdateWaferID Cassette1 8 XY12345
```

Wafer Map Commands

Map Commands
Die Commands
Subdie Commands
Cluster Commands
Bin Commands

Map Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands			
ConvertToAlphas	GetRectMapParams	ReadMapPosition	SetProductID
GetActiveLayer	GetWaferID	ReadMapPosition2	SetRectMapParams
GetCurrentBin	GetWaferInfo	ReadMapYield	SetWaferID
GetLotID	GetWaferMapMode	SaveMapFile	SetWaferMapMode
GetMapDims	GetWaferMapParams	SetActiveLayer	SetWaferMapParams
GetMapHome	GetWaferNum	SetCurrentBin	SetWaferMapParams2
GetMapOrientation	GetWaferTestAngle	SetLotID	SetWaferNum
GetMapRoute	GoToWaferHome	SetMapHome	SetWindowState
GetPreMappedDieInfo	NewWaferMap	SetMapOrientation	SetWaferTestAngle
GetProductID	OpenWaferMap	SetMapRoute	SyncMapHome

ConvertToAlphas

Description

Returns a string of the alpha column coordinates for the given range divided by spaces.

Parameter Input

Start

First column

Finish

Last column

Parameter Output

Alphas

Alpha Coordinates

Example

```
ConvertToAlphas 2 5
B C D E
```

GetActiveLayer

Description

Returns current active layer.

Parameter Output

Layer

C – Cluster layer
D – Die layer
S – Subdie layer

Example

```
GetActiveLayer
S
```

GetCurrentBin

Description

Returns the current bin in the application.

Parameter Output

Bin

Example

```
GetCurrentBin
15
```

GetLotID

Description

Returns the Lot ID for the current wafer.

Parameter Output

Lot ID

Example

```
GetLotID
80C24
```

GetMapDims

Description

Returns the parameters for the current wafer.

Parameter Output

MapType
W – Wafer
R – Rectangle

Xindex
Micrometer

Yindex
Micrometer

ColumnCount

RowCount

Example

```
GetMapDims 0: W 20000.0 20000.0 10 10
```

GetMapHome

Description

Reads the home die location of the WaferMap.

Parameter Output

Column
Column of the home die

Row
Row of the home die

Example

```
GetMapHome
5 4
```

GetMapOrientation

Description

Returns parameters of the map coordinate system. When 0, origin shift is from first die for XY. When 1, origin shift is from the corner based on orientation.

Parameter Input

UseOrientationCornerForShift (optional)
0 (default)
1

Parameter Output

Orientation
1 – Upper left
2 – Upper right
3 – Lower left
4 – Lower right

OriginShiftX
Shifted from first probe die in X (in coordinate system)

OriginShiftY
Shifted from first probe die in Y (in coordinate system)

UseAlphas

0 – Do not use alpha coordinates

1 – Use alpha coordinates

UseIOs

0 – Do not use “I” and “O” symbols in alpha coordinates

1 – Use “I” and “O” symbols in alpha coordinates

Example

```
GetMapOrientation 1
0: 3 2 8 0 1
```

GetMapRoute

Description

Returns the specified move path for the probe station inside the Wafer Map.

Parameter Input

MoveMode

B – Bidirectional testing (default)

S – Single-directional testing

C – Clockwise spiral (the center of the wafer is used as the starting point for the route)

R – Reverse clockwise spiral (the center of the wafer is used as the starting point for the route)

O – Route optimized

Start(column)

L – Left

R – Right

Start(row)

T – Top

B – Bottom

MoveDirection (for route simple)

H – Horizontal

V – Vertical

MinimizeMoveBy (for route optimized)

X

Y

B (both X and Y)

Examples

```
GetMapRoute
C L T H
```

```
GetMapRoute
R L T H
```

GetPreMappedDieInfo

Description

Returns actual (as measured during the premapping) x, y, z, and theta die positions. Also returns whether or not Z and Theta are being used.

This command is valid only when a wafer has been premapped in Spectrum.

Parameter Input

DieX (optional)

If not specified, this command returns values for the current die.

DieY (optional)

If not specified, this command returns values for the current die.

Parameter Output

UseZ

UseTheta

ActualX

ActualY

ActualZ

Theta

Example

```
GetPreMappedDieInfo 0
0 1 -39999.9 -59999.9 0.0 0.00018
```

Example

This example shows the behavior if a PreMapped Dies Table has not been loaded.

```
GetPreMappedDieInfo 715
Illegal parameters -- GetPreMappedDieInfo
```

GetProductID

Description

Displays the product ID for the current wafer.

Parameter Output

Product ID

Example

```
GetProductID
80C24
```

GetRectMapParams

Description

Gets the rectangle wafer map with specified parameters.

Parameter Output

X index

X index [µm]

Y index

Y index [µm]

DieAcross

Number of dies across (horizontal)

DieDown

Number of dies down (vertical)

Example

```
GetRectMapParams
0: 10000.0 10000.0 20 20
```

GetWaferID

Description

Returns the wafer ID for the current wafer.

Parameter Output

Wafer ID

Example

```
GetWaferID
12345678AB11.8B1F5
```

GetWaferInfo

Description

Returns a number of all units marked to test. Note that TestSites is a number of all sites for all dies.

Parameter Output

TestClusters

Total number of cluster on wafer.

TestDies

Total number of dies on wafer.

TestSites

Total number of subdies on wafer.

Example

```
GetWaferInfo
20 120 147
```

GetWaferMapMode

Description

Returns whether the application is in external mode or not.

Parameter Output

Mode

E – External mode (all controls in the application are disabled)

M – Controls are enabled

Example

```
GetWaferMapMode
M
```

GetWaferMapParams

Description

Returns the circle wafer parameters.

Parameter Output

WaferSize
Diameter [mm]

Xindex
Micrometer

Yindex
Micrometer

FlatLength
length [mm]

FlatAngle
0, 90, 180, 270 degrees

OffsetX
-99 to 99%

OffsetY
-99 to 99%

EdgeArea
area [mm]

Example

```
GetWaferMapParams
0: 200.0 20000.0 20000.0 84.4 180 7.5 -15.0 2.0
```

GetWaferMapParams2

Description

Returns the circle wafer parameters.

Parameter Output

WaferSize
Wafer diameter [mm]

Xindex
X index [µm]

Yindex
Y index [µm]

FlatLength
Length [mm]

FlatAngle
0, 90, 180, 270 degrees

OffsetX
X offset -99..99%

OffsetY
Y offset -99..99%

EdgeArea
Area [mm]

Example

```
GetWaferMapParams2
0: 200.0 20000.0 20000.0 84.4 180 1500.0 -3000.0 2.0
```

GetWaferNum

Description

Returns the wafer number for the current wafer.

Parameter Output

Wafer Number

Example

```
GetWaferNum
20
```

GetWaferTestAngle

Description

Returns the Wafer Test Angle.

Parameter Output

Wafer Test Angle

Example

```
GetWaferTestAngle
```

GoToWaferHome

Description

Moves the chuck to the wafer home position.

Example

```
GoToWaferHome
```

NewWaferMap

Description

Deletes current wafer map, sub dies and binning information; then creates new wafer map with defaults parameters.

Example

```
NewWaferMap
```

OpenWaferMap

Description

Opens a Wafer Map file.

Parameter Input

MapFile Name

Maximum 256 characters

Example

```
OpenWaferMap C:\Users\Public\Documents\Velox\Map\MyWaferMap.map
```

ReadMapPosition

Description

The ReadMapPosition command is maintained for legacy support.

Returns the actual Wafer Map chuck position. Returns the CurrentSubDie parameter as if it were a 1-based collection. To convert the returned CurrentSubDie parameter into the 0-based subdie collection, subtract 1.

NOTE

ReadMapPosition2 is the preferred method for reading Wafer Map chuck position.

Parameter Input

Pos (optional)

0 – Read current position (default)

1 – Read data for next die (if there is no next die, returns error 704)

-1 – Read data for previous die (if there is no previous die, returns error 704)

R (default)

Use Pos values described above. Pos is used as an absolute value:

1 – First die to probe

2 – Second die to probe

etc.

Parameter Output

Column

Row

xPosition

X position from home [µm]

yPosition

Y position from home [µm]

CurrentSubDie

Total SubDies

CurrentDie

Current die to probe

Total Dies

Total number of dies to probe

CurrentCluster

Current cluster to probe (counting from 1)

Total Clusters

Total number of clusters to probe (counting from 1)

Example

```
ReadMapPosition
4 7 -15000.0 45000.0 2 3 52 60 0 0
```

ReadMapPosition2

Description

Returns the actual Wafer Map chuck position. The SubDie collection is 0-based, the first value is 0, not 1. CurrentSubDie returns the currently-selected subdie (0-based) or -1 if no Subdie is currently selected.

The ReadMapPosition2 command is the preferred method for reading Wafer Map chuck position.

Parameter Input

Pos (optional)

0 – Read current position (default)

1 – Read data for next die (if there is no next die, returns error 704)

-1 – Read data for previous die (if there is no previous die, returns error 704)

R (default)

Use Pos values described above. Pos is used as an absolute value:

1 – First die to probe

2 – Second die to probe

etc.

Parameter Output

Column

Row

xPosition

X position from home [µm]

yPosition

Y position from home [µm

CurrentSubDie

Total SubDies

CurrentDie

Current die to probe

Total Dies

Total number of dies to probe

CurrentCluster

Current cluster to probe (counting from 1)

Total Clusters

Total number of clusters to probe (counting from 1)

Example

```
ReadMapPosition2
4 7 -15000.0 45000.0 1 3 52 60 0 0
```

ReadMapYield

Description

Returns the actual wafer map yield data information.

Parameter Output

Total Dies

Tested Dies

Passed

Failed

Example

```
ReadMapYield
123 100 50 50
```

SaveMapFile

Description

Returns the actual wafer map yield data information.

Parameter Input

FileName (optional)

If empty, saves under current

FileType (optional)

M – Native .map file (default)
S – .stv file
C – .csv file
K – .kla file format

Example

```
SaveMapFile waf23.map
```

SetActiveLayer

Description

Sets new active layer.

Parameter Input

Layer

C – Cluster layer
D – Die layer
S – Subdie layer

Example

```
SetActiveLayer D
```

SetCurrentBin

Description

Sets the current bin for the application. Second parameter allows to enable or disable “Mark with Bin” mode in the application.

Parameter Input

Bin

Binning mode status

0 – Disabled
1 – Enabled

Example

```
SetCurrenBin 15 1
```

SetLotID

Description

Specifies the ID for the current wafer.

Parameter Input

Lot ID

Example

```
SetLotID 8B1F5
0
```

SetMapHome

Description

If the command has no parameters it sets the current position as home position both for the wafer map and for the chuck. Otherwise, it changes only the wafer map home position using the given die coordinates.

Parameter Input

Column (optional)

Row (optional)

Example

```
SetMapHome
```

SetMapOrientation

Description

Sets up new map origin and new coordinate system after that.

Parameter Input

Orientation

1 – Upper left (default)
2 – Upper right
3 – Lower left
4 – Lower right

OriginShiftX

Shifted from first probe die in X (in coordinate system)

OriginShiftY

Shifted from first probe die in Y (in coordinate system)

UseAlphas

0 – Do not use alpha coordinates
1 – Use alpha coordinates

UseIOs

0 – Do not use “I” and “O” symbols in alpha coordinates
1 – Use “I” and “O” symbols in alpha coordinates

UseOrientationCornerForShift (optional)

Optional. Possible values 0 or 1. 0 is the default. When value is 0, Origin shift is from first die for XY. When value is 1, origin shift is from the corner based on orientation.

Example

```
SetMapOrientation 1 3 5 1 0 1
0:
```

SetMapRoute

Description

Specifies the Move Path for the probe station inside the Wafer Map. Optimization is done for the shortest move way for the station.

Parameter Input

MoveMode

B – Bidirectional testing (default)
S – Single-directional testing
C – Clockwise spiral (the center of the wafer is used as the starting point for the route)
R – Reverse clockwise spiral (the center of the wafer is used as the starting point for the route)
O – Route optimized

Start(column)

L – Left
R – Right

Start(row)

T – Top
B – Bottom

MoveDirection (for route simple)

H – Horizontal
V – Vertical

MinimizeMoveBy (for route optimized)

X
Y
B (both X and Y)

Examples

```
SetMapRoute B L T V
```

```
SetMapRoute O L T X
```

```
SetMapRoute C
```

```
SetMapRoute R
```

SetProductID

Description

Specifies the product ID for the current wafer.

Parameter Input

Product ID (optional)

Example

```
SetProductID 80C24
```

SetRectMapParams

Description

Creates the rectangle wafer with specified parameters.

Parameter Input

Xindex

Micrometer

Yindex

Micrometer

DieAcross

DieDown

Example

```
SetRectMapParams 10000 10000 10 10
```

SetWaferID

Description

Specifies the ID for the current wafer.

Parameter Input

Wafer ID

Example

```
SetWaferID 12345678AB11.8B1F5
```

SetWaferMapMode

Description

Enables or disables an external mode for the application. In the external mode, all controls are disabled. The application handles all its remote commands in both modes. Special interactive modes can also be turned on.

Parameter Input

Mode

E – External mode (all controls in the application are disabled)

M – Controls are enabled

Example

```
SetWaferMapMode E
0
```

SetWaferMapParams

Description

Creates the circle wafer with specified parameters. If parameter is omitted, then zero is used.

Parameter Input

WaferSize

Diameter [mm] (default 200)

Xindex
Micrometer. Less than WaferSize 1000 (default 10000).

Yindex
Micrometer. Less than WaferSize 1000 (default 10000).

FlatLength (optional)
Length [mm]. Less than WaferSize. If zero, the wafer is signed with a Notch (default 20).

FlatAngle (optional)
0, 90, 180, 270 degrees (default is 0)

OffsetX (optional)
-99 to 99% (default is 0)

OffsetY (optional)
-99 to 99% (default is 0)

EdgeArea (optional)
area [mm]. Less than 1/2 WaferSize (default is 0).

Example

```
SetWaferMapParams 300 10000 10000 10 90 0 0 1
```

SetWaferMapParams2

Description

Creates the circle wafer with specified parameters. If parameter is omitted, then zero is used.

Parameter Input

WaferSize
Diameter [mm] (default 200)

Xindex
Micrometer. Less than WaferSize 1000 (default 10000).

Yindex
Micrometer. Less than WaferSize 1000 (default 10000).

FlatLength (optional)
Length [mm]. Less than WaferSize. If zero, the wafer is signed with a Notch (default 20).

FlatAngle (optional)
0, 90, 180, 270 degrees (default is 0)

OffsetX (optional)
Micrometer (default is 0)

OffsetY (optional)
Micrometer (default is 0)

EdgeArea (optional)
area [mm]. Less than 1/2 WaferSize (default is 0).

Example

```
SetWaferMapParams2 300 23000 25000 0 90 1200 900 0
```

SetWaferNum

Description

Specifies the wafer number.

Parameter Input

Wafer Number

Example

```
SetWaferNum 20
```

SetWindowState

Description

Has sense in the application only (not for ActiveX controls). It shows or hides a window. All parameters are case-insensitive.

Parameter Input

State

S – Show
H – Hide

Window

Setup – Setup window
Bins – Bins window
Stat – Statistics window
DTbl – Dies table
STbl – Sites table
DMap – Die Map window
CTbl – Clusters Table
DInfo – Die Info window
AutoBin – Bin Auto Selection window
Prof – Profile window

Example

```
SetWindowState S STAT
```

SetWaferTestAngle

Description

This command sets the Wafer Test Angle. This is different from the notch angle and allows rotating the WaferMap.

Parameter Input

Angle

Example

```
SetWaferTestAngle 270
```

SyncMapHome

Description

SyncMapHome tries to find an appropriate position in the wafer for the current chuck position. X and Y describe the chuck position of the wafer center relative to the chuck zero position. If the wafer position is found, the command sets it as the wafer map home position and sets the current chuck position as the chuck home position.

If there is no corresponding wafer position, the command returns error 701. Command is used by AutoAlign for the BuildMap feature.

Parameter Input

X

Default is 0

Y

Default is 0

Example

```
SyncMapHome
```

Die Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands			
AssignMapBins	GetDieLabel	SetDieDataAsColRow	StepFailedBack
BinMapDie	GetDieLabelAsNum	SetDieDataAsNum	StepFailedForward
BinStepDie	GetDieResult	SetDieLabel	StepFirstDie
DoInkerRun	GetDieResultAsNum	SetDieLabelAsNum	StepNextDie
GetDieDataAsColRow	GetDieStatus	SetDieResult	StepNextDieOffset
GetDieDataAsNum	GetNumSelectedDies	SetDieResultAsNum	StepToDie
GetDieInfo	GetSelectedDieCoords	SetDieStatus	

AssignMapBins

Description

Assigns the pass or fail information to the actual bin value. Redundant to SetBinCode.

Parameter Input

Bins

P – Pass

F – Fail. No space, max. 255 ascii.

Example

```
AssignMapBins PPPPFFFF
```

BinMapDie

Description

Assigns the bin information to the wafer map at the current die, unless a row and column is specified. Inks the device if the real time inking option (of the wafer map) is enabled.

Parameter Input

Bin

0 to 255

Column (optional)

Row (optional)

Parameter Output

Column

Row

Example

```
BinMapDie 1
2 4
```

BinStepDie

Description

Bins the actual die and steps the chuck to the next selected die location of the wafer map (default) or steps to the specified Column and Row position, if these values are passed. In contrast to StepNextDie, the BinStepDie command will step from die to die while staying on the current subdie. It will not step through the subdies so it cannot be used for subdie stepping.

Parameter Input

Bin

0 to 255

Column (optional)

Row (optional)

SubDie (optional)

Parameter Output

Column

Row

SubDie

Total SubDies

Example

```
BinStepDie 3
-1 0 0 0
```

DoInkerRun

Description

Instructs the WaferMap to perform an ink run on the current wafer. Returns the number of dies which were inked during the inker run.

Parameter Output

Dies

Example

```
DoInkerRun
24
```

GetDieDataAsColRow

Description

Returns the Die Information in the Row Column format. If the Die Number is invalid, it returns an error. If the Die Column and Row are absent, it uses the current die on the wafer.

SEMI : <MD1> Used to send map data from equipment in row or column format

Parameter Input

Column

Row

Parameter Output

DieNumber

Column

Row

Bincode

0 to 255

Result

Measurement result

Example

```
GetDieDataAsColRow 2 5
13 2 5 2 100.3V
```

GetDieDataAsNum

Description

Returns the Die Information in the Row Column format. If the Die Number is invalid it returns an error. If the Die Number is absent it uses the current die on the wafer.

SEMI : <MD1> Used to send map data from equipment in row or column format

Parameter Input

DieNumber (optional)

Parameter Output

DieNumber

Die number returned.

Column

Row

Bincode

0 to 255

Result

Measurement result

Example

```
GetDieDataAsNum 25
25 4 5 2 100.3V
```

GetDieInfo

Description

Returns a number of sites marked to test in the given die.

Parameter Input

Die (optional)

Current by default

Parameter Output

TestSites

Number of test sites.

Example

```
GetDieInfo
6
```

GetDieLabel

Description

Returns a label from the wafer map at the current die, unless a row and column is specified.

Parameter Input

Column (optional)

Default is 0

Row (optional)

Default is 0

Parameter Output

Label

The label for the specified die (maximum of 256 characters)

Example

```
GetDieLabel
"This is my die label serial # 123456"
```

GetDieLabelAsNum

Description

Returns a label from the wafer map at the current die, unless a die number is specified.

Parameter Input

Die (optional)

Default is the current die.

Parameter Output

Label

The label for the specified die (maximum of 256 characters)

Example

```
GetDieLabelAsNum
"This is my die label serial # 123456"
```

GetDieResult

Description

Returns a measurement result into the wafer map at the current die, unless a row and column are specified.

Parameter Input

Column (optional)

Row (optional)

Parameter Output

Result

No spaces (maximum 256 characters)

Example

```
GetDieResult 1 2
: 300.0V
```

GetDieResultAsNum

Description

Returns a measurement result from the wafer map at the current die, unless a die number is specified.

Parameter Input

Die (optional)

Parameter Output

Result

No spaces (maximum 256 characters)

Example

```
GetDieResultAsNum 1
: 300.0V
```

GetDieStatus

Description

Returns the die status.

Parameter Input

Column

Row

Parameter Output

Status

X – Die is invalid or does not exist
P – Die is to be probed
I – Die is to be inked only, and not to be probed
V – Die is valid but should not be probed

Example

```
GetDieStatus 3 4
X
```

GetNumSelectedDies

Description

Gets the number of dies in the map which are selected for probing.

Parameter Output

NumSelectedDies

Example

```
GetNumSelectedDies
486
```

GetSelectedDieCoords

Description

Given the index of a selected die in the range [1..NumSelectedDies], returns the column and row indices for the selected die.

Parameter Input

SelectedDieIndex

Parameter Output

Die Column Index

Die Row Index

Example

```
GetSelectedDieCoords 144
12 8
```

SetDieDataAsColRow

Description

Sets the Data in the Row Column format. If the Die is invalid it returns an error.

SEMI : <MD1> Used to send map data from equipment in row or column format

Parameter Input

Column

Row

Bincode

0 to 255

Result (optional)

No spaces (maximum 256 characters)

Example

```
SetDieDataColRow 4 5 2 100.3V
```

Bin Auto Selection

If auto binning is used with SetDieDataAsColRow, the die Results must be compatible with the Bin Auto Selection.

For instance, if the “Result string format” field, on the WaferMap, Bin Auto Selection tab, is defined as follows:

val1=%a,val2=%b,val3=%c,val4=%d

then the die (or subdie) “result” string should be of the following form:

val1=99,val2=4.0,val3=400,val4=58

This would yield a SetDieResult SCI command as follows (when defaulting to the current die):

SetDieResult val1=99,val2=4.0,val3=400,val4=58

SetDieDataAsNum

Description

Sets the Die Information. If the Die Number is invalid it returns an error.

SEMI : <MD1> Used to send map data from equipment in row or column format

Parameter Input

DieNumber
Bincode 0 to 255
Result (optional) No spaces (maximum 256 characters)
Example `SetDieDataAsNum 25 2 100.3V`

Bin Auto Selection

If auto binning is used with SetDieDataAsNum, the die Results must be compatible with the Bin Auto Selection.

For instance, if the “Result string format” field, on the WaferMap, Bin Auto Selection tab, is defined as follows:

val1=%a,val2=%b,val3=%c,val4=%d

then the die (or subdie) “result” string should be of the following form:

val1=99,val2=4.0,val3=400,val4=58

This would yield a SetDieResult SCI command as follows (when defaulting to the current die):

SetDieResult val1=99,val2=4.0,val3=400,val4=58

SetDieLabel

Description Assigns a label in the wafer map at the current die, unless a row and column is specified.
Parameter Input
Label (optional) Up to 256 characters
Column (optional)
Row (optional)
Example `SetDieLabel “This is my die label serial # 123456” 6 2`

SetDieLabelAsNum

Description Assigns a label in the wafer map at the current die, unless a die number is specified.
Parameter Input
Label (optional) Up to 256 characters
Die (optional)
Example `SetDieLabelasNum “This is my die label serial # 123456” 6`

SetDieResult

Description

Assigns a measurement result in the wafer map at the current die, unless a row and column is specified.

Parameter Input

Result (optional)

No spaces (maximum 256 characters)

Column (optional)

Row (optional)

Example

```
SetDieResult 2345.344 1 2
```

Bin Auto Selection

If auto binning is used with SetDieResult, the die Results must be compatible with the Bin Auto Selection.

For instance, if the “Result string format” field, on the WaferMap, Bin Auto Selection tab, is defined as follows:

val1=%a,val2=%b,val3=%c,val4=%d

then the die (or subdie) “result” string should be of the following form:

val1=99,val2=4.0,val3=400,val4=58

This would yield a SetDieResult SCI command as follows (when defaulting to the current die):

SetDieResult val1=99,val2=4.0,val3=400,val4=58

SetDieResultAsNum

Description

Assigns a measurement result in the wafer map at the current die, unless a die number is specified.

Parameter Input

Result (optional)

No spaces (maximum 256 characters)

Die (optional)

Example

```
SetDieResultAsNum 2345.344 1
```

SetDieStatus

Description

Sets the die status.

Parameter Input

Column

Row

Status

X – Die is invalid or does not exist

P – Die is to be probed

I — Die is to be inked only, and not to be probed

V – Die is valid but should not be probed (skip).

NOTE

When changing die status from P to V during a test, the die status will not be changed during the current test. It is deferred until the "next test." When the next test is started, by sending StepFirstDie, the die status will be updated, and the WaferMap route will be recalculated.

Example

```
SetDieStatus 3 4
```

StepFailedBack

Description

Steps to the last known good die. That is, steps backwards in the route to the last die that was binned with a status of Pass. Returns the number of dies that it stepped back (i.e., over).

Parameter Output

Failed Die

Example

```
StepFailedBack
7
```

StepFailedForward

Description

Steps to the next untested die. That is, steps forwards in the route to the next die that has not been binned. Returns the number of dies that it stepped forward (i.e., over).

Parameter Output

Failed Die

Example

```
StepFailedForward
7
```

StepFirstDie

Description

Steps the chuck to the first die of the wafer map. Returns the row and column number of the actual die location after the move is completed. Velox switches to remote control if the "Disable Velox" flag is set in the external options window.

Parameter Input

ClearBins (optional)

0 – No

1 – Yes (default)

If yes, all binning data will be cleared from dies.

RecalcRoute (optional)

0 – No

1 – Yes (default)

If yes, the route will be recalculated. All dies marked to skip during the last test (marked to skip with SetDieStatus) will be eliminated from the route.

Parameter Output

Column

Row

SubDie

Total SubDies

Example

```
StepFirstDie
0 0 1 5
```

StepNextDie

Description

Steps the chuck to the specified x,y die location of the wafer map. If no command data is passed, the chuck automatically steps to the next logical wafer map die location. Returns the row and column number of the actual die location after the move is completed. In addition, the SubDie location, and total number of SubDies is also returned.

If Site (i.e. SubDie) is -1, the chuck will move to SubDie 0, on the next die; in this case, the first 2 parameters are ignored.

When using this command from shared code, use SendWithoutParameter().

Parameter Input

Column (optional)

Row (optional)

SubDie (optional)

If this value is -1, the chuck will move to SubDie 0 on the next die; in this case, the first two parameters are ignored.

Parameter Output

Column

Row

SubDie

Total SubDies

Example

```
StepNextDie
0 0 1 5
```

StepNextDieOffset

Description

Moves to a user-specified offset within a die. The optional Column and Row parameters can be used to specify the die. If they are omitted, WaferMap moves to the specified offset within the next selected die.

Parameter Input

X offset (microns)

X offset in microns. Default is 0.

Y offset (microns)
Y offset in microns. Default is 0.

Column (optional)

Row (optional)

Parameter Output

Column
Column of the new current die.

Row
Row of the new current die.

SubDie

Total SubDies

Example

```
StepNextDieOffset -188.5 127.0
13 8
```

StepToDie

Description

Steps the chuck to the die location specified by the die number of the wafer map. If no command data is passed, the chuck automatically steps to the next logical wafer map die location. Returns the row and column number of the actual die location after the move is completed. In addition, the SubDie location, and total number of SubDies is also returned.

Parameter Input

DieNumber
Die number of the target die. Default is -1.

Site (optional)
Subdie on the target die. Default is 0.

Parameter Output

DieColumn
Column (X) of the new current die.

DieRow
Row (Y) of the new current die.

CurrentSite
Current Subdie number.

TotalSubdies
Total number of subdies.

Example

```
StepToDie
```

Subdie Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands

AddSubDie	GetDieMapResultAsNum	GetSubDieLabelAsNum	SetSubDieDataAsColRow
BinSubDie	GetDieRefPoint	GetSubDieStatus	SetSubDieDataAsNum
DeleteAllSubDie	GetSubDieData	SetDieMapResult	SetSubDieLabel
DeleteSubDie	GetSubDieDataAsColRow	SetDieMapResultAsNum	SetSubDieLabelAsNum
DeleteSubDie2	GetSubDieDataAsNum	SetDieRefPoint	SetSubDieStatus
GetDieMapResult	GetSubDieLabel	SetSubDieData	StepNextSubDie

AddSubDie

Description

Adds new subdie at the end of the table.

Parameter Input

X

The subdie's X position

Y

The subdie's Y position

Label (optional)

Up to 256 characters

Parameter Output

Count

Number of subdies available for deleting

Example

```
AddSubDie -500 133.33 Test
3
```

BinSubDie

Description

Assigns the bin information for the current subdie in the current die, unless a subdie, row, and column are specified. The SubDie collection is 0-based, the first value is 0, not 1.

Parameter Input

Bin

0 to 255

Column (optional)

Row (optional)

SubDie (optional)

Parameter Output

Column

Row

SubDie

Total SubDies

Example

```
BinSubDie 15 4 7 3
4 7 3 5
```

DeleteAllSubDie

Description

Deletes all subdies for all dies.

Example

```
DeleteAllSubDie
```

DeleteSubDie

Description

The DeleteSubDie command is maintained for legacy support.

Deletes selected subdie. Accepts a SubDie index as if it were a 1-based collection. To convert the parameter from the 0-based subdie collection, add 1. If 0 is passed in as the subdie index, WaferMap will delete all subdies for all dies.

- DeleteSubDie2 is the preferred method for deleting a selected subdie.
- DeleteAllSubDie is the preferred method for deleting all subdies for all dies.

Parameter Input

Index

The subdie's index

Parameter Output

Count

Number of subdies available for deleting

Example

```
DeleteSubDie 3
7
```

DeleteSubDie2

Description

Deletes selected subdie. The SubDie collection is 0-based, the first value is 0, not 1.

This command is the preferred method for deleting a selected subdie.

Parameter Input

Index

The subdie's index

Parameter Output

Count

Number of subdies available for deleting

Example

```
DeleteSubDie2 2
7
```

GetDieMapResult

Description

Returns the subdie measurement result for the specified subdie, at the specified die. All parameters must be supplied.

Parameter Input

Column (optional)
Default is 0.

Row (optional)
Default is 0.

SubDie (optional)
Default is 0.

Parameter Output

Result
No spaces (maximum 256 characters)

Example

```
GetDieMapResult 1 1 2
```

GetDieMapResultAsNum

Description

Return a measurement result from the wafer map at the current die and at the current subdie, unless a subdie, and die number are specified. The SubDie collection is 0-based, the first value is 0, not 1.

Parameter Input

Die (optional)

SubDie (optional)

Parameter Output

Result
No spaces (maximum 256 characters)

Example

```
GetDieMapResultAsNum 1 1
```

GetDieRefPoint

Description

Returns current reference point for the die. For details see SetDieRefPoint command.

Parameter Output

RefX
X coordinate

RefY
Y coordinate

Example

```
GetDieRefPoint
-100 200
```

GetSubDieData

Description

Returns the information of a SubDie. The SubDie collection is 0-based, the first value is 0, not 1.

Parameter Input

SubDie

Parameter Output

SubDie

X

Relative position inside the die (in micrometers)

Y

Relative position inside the die (in micrometers)

Label

Example

```
GetSubDieData 3
3 456.5 433.3 Ground
```

GetSubDieDataAsColRow

Description

Returns the Die information in the Row Column format. Returns an error if the Die number is invalid. The SubDie collection is 0-based, the first value is 0, not 1.

SEMI : <MD1> Used to send map data from equipment in row or column format

Parameter Input

Column

Row

SubDie

Parameter Output

DieNumber

Column

Row

SubDie

Bincode

0 to 255

Result

Measurement result

Example

```
GetSubDieDataAsRowCol 4 5 3
25 4 5 3 2 200.8V
```

Bin Auto Selection

If auto binning is used with SetSubDieDataAsColRow, the subdie Results must be compatible with the Bin Auto Selection.

For instance, if the “Result string format” field, on the WaferMap, Bin Auto Selection tab, is defined as follows:

val1=%a,val2=%b,val3=%c,val4=%d

then the die (or subdie) “result” string should be of the following form:

val1=99,val2=4.0,val3=400,val4=58

This would yield a SetDieResult SCI command as follows (when defaulting to the current die):

SetDieResult val1=99,val2=4.0,val3=400,val4=58

GetSubDieDataAsNum

Description

Returns the Die information in the Row Column format. Returns an error if the Die number is invalid. The SubDie collection is 0-based, the first value is 0, not 1.

SEMI : <MD1> Used to send map data from equipment in row or column format

Parameter Input

DieNumber

SubDie

Parameter Output

DieNumber

Column

Row

SubDie

Bincode

0 to 255

Result

Measurement result

Example

```
GetSubDieDataAsNum 25 10
25 4 5 10 2 200.8V
```

GetSubDieLabel

Description

Return a label from the wafer map at the current die, unless a row and column are specified. The SubDie collection is 0-based, the first value is 0, not 1.

Parameter Input

Column

Row

Subdie

Parameter Output

Label

The label for the specified subdie (maximum of 256 characters)

Example

```
GetSubDieLabel
"This is my subdie label serial # 123456"
```

GetSubDieLabelAsNum

Description

Return a label from the wafer map the current die at the current subdie, unless a subdie, and die number are specified. The SubDie collection is 0-based, the first value is 0, not 1.

Parameter Input

Die (optional)

SubDie (optional)

Parameter Output

Label

The label for the specified subdie (maximum of 256 characters)

Example

```
GetSubDieLabelAsNum
"This is my subdie label serial # 123456"
```

GetSubDieStatus

Description

Returns the SubDie status. The SubDie collection is 0-based, the first value is 0, not 1.

Note: *Use 0 for getting status of Die Origin subdie.*

Parameter Input

SubDie

Parameter Output

Status

E – Enabled

D – Disabled

Example

```
GetSubDieStatus 0
D
```

SetDieMapResult

Description

Assigns a measurement result in the wafer die map at the current die at the current subdie, unless a subdie, row, and column are specified. The SubDie collection is 0-based, the first value is 0, not 1.

Parameter Input

Result (optional)

No spaces (maximum 256 characters)

Column (optional)

Row (optional)

SubDie (optional)

Example

```
SetDieMapResult 2345.344 1 2 0
```

Bin Auto Selection

If auto binning is used with SetDieMapResult, the subdie Results must be compatible with the Bin Auto Selection.

For instance, if the "Result string format" field, on the WaferMap, Bin Auto Selection tab, is defined as follows:

val1=%a,val2=%b,val3=%c,val4=%d

then the die (or subdie) "result" string should be of the following form:

val1=99,val2=4.0,val3=400,val4=58

This would yield a SetDieResult SCI command as follows (when defaulting to the current die):

SetDieResult val1=99,val2=4.0,val3=400,val4=58

SetDieMapResultAsNum

Description

Assigns a measurement result in the wafer die map at the current die at the current subdie, unless a subdie and die number are specified. The SubDie collection is 0-based, the first value is 0, not 1.

Parameter Input

Result (optional)

No spaces (maximum 256 characters)

Die (optional)

SubDie (optional)

Example

```
SetDieMapResult 2345.344 1 2
```

SetDieRefPoint

Description

Sets a new reference point for the die. This point is NOT used for die stepping, nor for subdie stepping. When using the "Chuck Position from Reference" WaferMap GUI setting, the Die Reference Point can be set to be more representative of the actual reference location within the die.

A setting of (0.0, 0.0) is defined as the upper left corner of the die. All subdie coordinates are relative to upper left corner of the die.

Parameter Input

RefX

X coordinate (in micrometers)

RefY

Y coordinate (in micrometers)

Example

```
SetDieRefPoint -100 200
```

SetSubDieData

Description

Sets up SubDie data. The SubDie collection is 0-based, the first value is 0, not 1.

Parameter Input

SubDie

X

Relative position inside the die

Y

Relative position inside the die

Label (optional)

Maximum 256 characters

Example

```
SetSubDieData 1 100 300
```

SetSubDieDataAsColRow

Description

Sets the SubDie information in the Row Column format. Returns an error if the Die number is invalid. The SubDie collection is 0-based, the first value is 0, not 1.

SEMI : <MD1> Used to send map data from equipment in row or column format

Parameter Input

Column

Row

SubDie

Bincode

0 to 255

Result (optional)

No spaces (maximum 256 characters)

Example

```
SetSubDieDataAsColRow 10 4 5 2 90.3V
```

Bin Auto Selection

If auto binning is used with SetSubDieDataAsColRow, the subdie Results must be compatible with the Bin Auto Selection.

For instance, if the “Result string format” field, on the WaferMap, Bin Auto Selection tab, is defined as follows:

val1=%a,val2=%b,val3=%c,val4=%d

then the die (or subdie) “result” string should be of the following form:

val1=99,val2=4.0,val3=400,val4=58

This would yield a SetDieResult SCI command as follows (when defaulting to the current die):

SetDieResult val1=99,val2=4.0,val3=400,val4=58

SetSubDieDataAsNum

Description

Sets the SubDie information in the Row Column format. Returns an error if the Die number is invalid. The SubDie collection is 0-based, the first value is 0, not 1.

Parameter Input

DieNumber

SubDie

Bincode

Result

No spaces (maximum 256 characters)

Example

```
SetSubDieDataAsNum 25 10 4 5 2 90.3V
```

Bin Auto Selection

If auto binning is used with SetSubDieDataAsNum, the subdie Results must be compatible with the Bin Auto Selection.

For instance, if the “Result string format” field, on the WaferMap, Bin Auto Selection tab, is defined as follows:

val1=%a,val2=%b,val3=%c,val4=%d

then the die (or subdie) “result” string should be of the following form:

val1=99,val2=4.0,val3=400,val4=58

This would yield a SetDieResult SCI command as follows (when defaulting to the current die):

SetDieResult val1=99,val2=4.0,val3=400,val4=58

SetSubDieLabel

Description

Sets a label into the wafer map at the current subdie, unless a row and column and subdie are specified.

Parameter Input

Label

The label for the specified subdie (maximum of 256 characters)

Column

Row

Subdie

Example

```
SetSubDieLabel “This is my subdie label serial # 123456” 3 2 1
```

SetSubDieLabelAsNum

Description

Sets a label into the wafer map at the current subdie, unless a die and subdie are specified.

Parameter Input

Label

The label for the specified subdie (maximum of 256 characters)

Die (optional)

Subdie (optional)

Example

```
SetSubDieLabelAsNum “This is my subdie label serial # 123456”
```

SetSubDieStatus

Description

Sets the SubDie status. The SubDie collection is 0-based, the first value is 0, not 1.

Note: *Use 0 for* setting *status of Die Origin subdie.*

Parameter Input

SubDie

Status

E – Enable

D – Disable

Example

```
SetSubDieStatus 2 D
```

StepNextSubDie

Description

Steps the chuck to the specified SubDie number relative to the current die origin of the wafer map. Returns the SubDie number of the actual die location after the move is completed, and the total number of subdies.

Parameter Input

SubDie (optional)

Default is 0

Parameter Output

SubDie

Current subdie

Total SubDies

Total number of subdies (or index of the last subdie)

Example

```
StepNextSubDie
4 30
```

Cluster Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands		
GetClusterDieStatus	GetSelectedClusterCoords	StepFailedClusterBack
GetClusterInfo	ReadClusterPosition	StepFailedClusterForward
GetClusterParams	SetClusterDieStatus	StepFirstCluster
GetNumSelectedClusters	SetClusterParams	StepNextCluster

GetClusterDieStatus

Description

Gets the die status in the cluster. Die coordinates are internal for the cluster using the wafer map coordinate system. (0,0) die in the cluster is left top corner (whether this die exists or not).

Parameter Input

Cluster Column

Cluster Row

Die Column

Die Row

Parameter Output

Status

X – Die is invalid or does not exist

P – Die is to be probed

I – Die is to be inked only, and not to be probed

V – Die is valid but should not be probed

Example

```
GetClusterDieStatus 1 2 0 3
X
```

GetClusterInfo

Description

Returns a number of all dies and sites marked to test in the given cluster.

Parameter Input

Cluster (optional)

Cluster number. Current, by default

Parameter Output

TestDies

TestSites

Number of all sites for all dies in the given cluster.

Example

```
GetClusterInfo 3
6 15
```

GetClusterParams

Description

Returns current clusters parameters. The stepping commands only step to clusters that are in the route.

Parameter Output

UseClusters

0 – No

1 – Yes

ClusterWidth

(in dies)

ClusterHeight

(in dies)

TestIncomplete

0 – No, incomplete clusters are not included in the route, and therefore are not testable.

1 – Yes, incomplete clusters are included in the route, and therefore are testable.

Example

```
GetClusterParams
1 3 1 0
```

GetNumSelectedClusters

Description

Gets the number of clusters in the map which are selected for probing.

Parameter Output

NumSelectedClusters

Example

```
GetNumSelectedClusters
57
```

GetSelectedClusterCoords

Description

Given the index of a selected cluster in the range [1..NumSelectedClusters], returns the column and row indices for the selected cluster.

Parameter Input

SelectedClusterIndex

Parameter Output

Cluster Column Index

Cluster Row Index

Example

```
GetSelectedClusterCoords 35
3 5
```

ReadClusterPosition

Description

Returns the actual wafer map cluster position. If the cluster probing is disabled, the command will assume a cluster of size 1x1.

Parameter Input

Pos

0 – Read current position (default)

1 – Read data for next die (if no dies exist for this value, returns error 716)

-1 – Read data for previous die (if no dies exist for this value, returns error 716)

Parameter Output

Cluster Column

Cluster Row

Cluster Index

Die Column

Die Row

Die Index

Cluster Width

Cluster Height

Enabled Dies

0 – Die is disabled

1 – Die is enabled

Example

```
ReadClusterPosition
2 1 10 5 4 72 2 3 111111
```

SetClusterDieStatus

Description

Sets the die status in the cluster. Die coordinates are internal for the cluster using the wafer map coordinate system. (0,0) die in the cluster is left top corner (whether this die exists or not).

Parameter Input

Cluster Column

Cluster Row

Die Column

Die Row

Status

X – Die is invalid or does not exist

P – Die is to be probed

I – Die is to be inked only, and not to be probed

V – Die is valid but should not be probed

Example

```
SetClusterDieStatus 3 4 2 2 P
```

SetClusterParams

Description

Specifies the clusters parameters.

Parameter Input

UseClusters

0 – No

1 – Yes

ClusterWidth

In dies

ClusterHeight

In dies

TestIncomplete

0 – No, incomplete clusters are not included in the route, and therefore are not testable.

1 – Yes, incomplete clusters are included in the route, and therefore are testable.

Example

```
SetClusterParams 1 3 1 0
```

StepFailedClusterBack

Description

Steps to the last known good cluster (all dies within the cluster binned with a status of Pass). That is, steps backwards in the route to the last cluster that was binned with a status of Pass. Returns the number of clusters that it stepped back (i.e., over).

Parameter Output

Failed Cluster

Number of consecutive failed clusters.

Example

```
StepFailedClusterBack
7
```

StepFailedClusterForward

Description

Steps to the next untested cluster. That is, steps forwards in the route to the next cluster that has at least 1 die that has not been binned. Returns the number of clusters that it stepped forward (all dies within the cluster have been binned).

Parameter Output

Failed Cluster

Number of consecutive failed clusters.

Example

```
StepFailedClusterForward
7
```

StepFirstCluster

Description

Steps the chuck to the first die cluster of the cluster map. If clusters are not defined, the first Die is used. Returns the row and column information of the actual cluster and die location after the move is completed. Switches to remote control if the “Disable Velox” flag is set in the external options window.

Parameter Input

ClearBins (optional)

0 – No

1 – Yes (default)

If yes, all binning data will be cleared from dies.

RecalcRoute (optional)

0 – No

1 – Yes (default)

If yes, the route will be recalculated. All dies marked to skip during the last test (marked to skip with SetDieStatus) will be eliminated from the route.

Parameter Output

ClusterColumn

ClusterRow

ClusterNum

ClusterIncomplete 0 – No 1 – Yes
DieColumn
DieRow
DieNum

Example

```
StepFirstCluster
0 0 1 0 1 5 4
```

StepNextCluster

Description

Steps the chuck to the specified x,y cluster die location of the cluster map. If no command data is passed, the chuck automatically steps to the next logical cluster map location. If clusters are not defined, the Die information is used. Returns the row and column number of the actual die location after the move is completed.

Parameter Input
ClusterColumn (optional)
ClusterRow (optional)

Parameter Output
ClusterColumn
ClusterRow
ClusterNum
ClusterIncomplete 0 – No 1 – Yes
DieColumn
DieRow
DieNum

Example

```
StepNextCluster 1 2
1 2 2 1 1 5 9
```

Bin Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands			
ClearAllBins	GetBinTableSize	SaveBinCodeTable	SetBinTableSize
GetBinCode	OpenBinCodeTable	SetBinCode	

ClearAllBins

Description

Clears all binning data in the wafer map.

Example

```
ClearAllBins
```

GetBinCode

Description

Returns the bin information for the specified bin code.

Parameter Input

Bincode

0 to 255 (default is 0)

Parameter Output

Char

Character representation (2 characters)

Color

Microsoft Windows color representation

Status

P – Pass

F – Fail

Inker1

0 – Inker disabled

1 – Inker enabled

Inker2

0 – Inker disabled

1 – Inker enabled

Example

```
GetBinCode 2
AA 187 P 0 1
```

GetBinTableSize

Description

Gets size of the bin code table.

Parameter Output

Size

Size of the bin code table.

Example

```
GetBinTableSize
32
```

OpenBinCodeTable

Description

Opens a Bin Code Table file (.bct) with the specified name.

Parameter Input

Filename

Maximum 256 characters

Example

```
OpenBinCodeTable c:\bct01.bct
```

SaveBinCodeTable

Description

Saves bin code table to a file with the specified name.

Parameter Input

Filename

Maximum 256 characters

Example

```
SaveBinCodeTable bct01.bct
```

SetBinCode

Description

Appends the Bin information to a bin code.

Parameter Input

Bincode

0 to 255

Chars

Character representation (2 characters)

Color

Microsoft Windows color representation

Status

P – Pass
F – Fail

Inker1

0 – Inker disabled
1 – Inker enabled

Inker2

0 – Inker disabled
1 – Inker enabled

Example

```
SetBinCode 15 ba 0x0040C0 F 0
```

SetBinTableSize

Description

Sets size of the bin code table. This is the size of bins used in statistics and binning.

Parameter Input

Size

Size of the bin code table.

Example

```
SetBinTableSize 32
```

Profiling Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands		
DoWaferProfiling	GetWaferProfilingStatus	StartWaferProfiling
GetWaferProfileOptions	SetWaferProfileOptions	StopWaferProfiling

DoWaferProfiling

Description

Profiles the wafer and returns a status of the profiling as an error code.

Example

```
DoWaferProfiling
```

GetWaferProfileOptions

Description

Returns predefined settings of the WaferMap profiling options.

Parameter Output

Profile Sensor

E – Edge Sensor

F – Find Focus Algorithm

Search Speed

% – Speed of Z Axis (important for Edge Sensor only)

Tolerance Gap

µm – Tolerance for the Z Axis (important for Edge Sensor only)

Success Ratio

% – Number of successfully profiled points (percentage) in determining successful profiling

Profiling Distance X

X value to add to the profile point during the profiling

Profiling Distance Y

Y value to add to the profile point during the profiling

Example

```
GetWaferProfileOptions
E 50 10 75 0 0
```

GetWaferProfilingStatus

Description

Returns status of the profiling process. If this status is true, the process is started.

Parameter Output

Status

0 – Profiling is not running

1 – Profiling is currently running

Example

```
GetWaferProfilingStatus
0
```

SetWaferProfileOptions

Description

Sets some WaferMap profiling options remotely.

Parameter Input

Profile Sensor

E – Edge Sensor

F – Find Focus Algorithm

Search Speed

% – Speed of Z Axis (important for Edge Sensor only)

Tolerance Gap

µm – Tolerance for the Z Axis (important for Edge Sensor only)

Success Ratio

% – Number of successfully profiled points (percentage) in determining successful profiling

Profiling Distance X (optional)

X value to add to the profile point during the profiling

Profiling Distance Y (optional)

Y value to add to the profile point during the profiling

Example

```
SetWaferProfileOptions E 50 10 75
```

StartWaferProfiling

Description

Starts the profiling process. To determine the current profiling state, use GetWaferProfilingStatus command.

Parameter Input

DoContinue

0 – Start new profiling (default)

1 – Continue previously-stopped profiling

Example

```
StartWaferProfiling
```

StopWaferProfiling

Description

Stops the profiling process.

Example

```
StopWaferProfiling
```

Chapter

9 Kernel Commands

The following command sets are included in this section:

- Alignment Commands
- Chuck Commands
- Positioner Commands
- Scope Commands
- AUX Site Commands
- Theta Commands
- Turret Commands
- System Commands
- Offset Commands
- Setup Commands
- Temperature Chuck Commands
- Notifications

Alignment Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands		
AlignCardTheta	AlignScopeTheta	ReadProbeTheta
AlignChuckTheta	ReadCardTheta	ReadScopeTheta
AlignProbeTheta	ReadChuckTheta	

AlignCardTheta

Description

This command causes a rotation of the chuck coordinate system to align the chuck X,Y axis to the probe card. The polarity of the data determines a left or a right rotation of the chuck coordinate system.

Parameter Input

Angle (required)

Unit (optional)

D – Degree (default)

A – Radian

Reference

R – Relative (default)

Z – Zero

Example

```
AlignCardTheta 4.2 D R
```

AlignChuckTheta

Description

This command causes a chuck Theta axis rotation to align the wafer to the chuck X,Y movements. It can be used to perform a two point alignment with P1(X1,Y1) and P2(X2,Y2) as two points on a wafer street line.

Calculation of X and Y distances:

X Distance = X2 - X1;

Y Distance = Y2 - Y1;

The units of the distances are not important – but both distances should use the same unit.

If one or both distances are zero, the command does not return an error but ignores this alignment instead.

Parameter Input

X Distance

Y Distance

Reference

R – Relative (default)

Z – Zero

Example

```
AlignChuckTheta 14355.8 -433.2 R
```

AlignProbeTheta

Description

This command causes a rotation of the coordinate system of given probe to align the probe to the chuck X,Y axis or/and to the wafer alignment. It can be used to perform a two point alignment with the points P1(X1,Y1) and P2(X2,Y2). Calculation of Y and Y distances:

X Distance = X2 - X1;

Y Distance = Y2 - Y1;

The unit if the distances are not important – but both distances should use the same unit.

Parameter Input

Probe number (required)

1 – Positioner 1

2 – Positioner 2

3 – Positioner 3

4 – Positioner 4

5 – Positioner 5

6 – Positioner 6

X Distance (required)

Y Distance (required)

Reference (optional)

R – Relative (default)

Z – Zero

Parameter Output

Probe number

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
AlignProbeTheta 1 14355.8 -433.2 R
1
```

AlignScopeTheta

Description

This command causes a virtual rotation of the scope coordinate system to align the scope to the chuck X,Y axis or/and to the wafer alignment. It can be used to perform a two point alignment with the points P1(X1,Y1) and P2(X2,Y2). Calculation of X and Y distances:

X Distance = X2 - X1;

Y Distance = Y2 - Y1;

The unit of the distances is not important – but both distances should use the same unit.

Parameter Input

X Distance (required)

Y Distance (required)

Reference (optional)

R – Relative (default)
Z – Zero

Example

```
AlignScopeTheta 14355.8 -433.2 R
```

ReadCardTheta

Description

Returns the angle between the chuck coordinate system and the probe card coordinate system.

Parameter Input

Unit (optional)

D – Degree (default)
A – Radian

Parameter Output

Angle

Example

```
ReadCardTheta D
4.2
```

ReadChuckTheta

Description

Returns the current chuck alignment angle which is identical to the current value of theta rotation.

Parameter Input

Unit (optional)

D – Degree (default)
A – Radian

Parameter Output

Angle

Example

```
ReadChuckTheta D
4.2
```

ReadProbeTheta

Description

Returns the actual positioner's alignment angle, which is the angle between the chuck coordinate system and the positioner coordinate system.

Parameter Input

Probe number (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Unit (optional)

D – Degree (default)
A – Radian

Parameter Output

Probe number

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Angle

Example

```
ReadProbeTheta 1 D
1 4.2
```

ReadScopeTheta

Description

Returns the scope's alignment angle, which is the angle between the chuck coordinate system and the scope coordinate system.

Parameter Input

Unit (optional)

D – Degree (default)

A – Radian

Parameter Output

Angle

Example

```
ReadScopeTheta D
4.2
```

Chuck Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands			
InitChuck	MoveChuckSeparation	ReadChuckIndex	SetChuckHome
MoveChuck	MoveChuckSubsite	ReadChuckPosition	SetChuckIndex
MoveChuckAlign	MoveChuckTransfer	ReadChuckStatus	SetChuckMode
MoveChuckContact	MoveChuckVelocity	ReadChuckThermoScale	SetChuckThermoScale
MoveChuckIndex	MoveChuckZ	ReadChuckThermoValue	SetChuckThermoValue
MoveChuckLift	MoveChuckZSafe	SearchChuckContact	StopChuckMovement
MoveChuckLoad	ReadChuckHeights	SetChuckHeight	

InitChuck

Description

Initializes the chuck stage and resets current coordinate system. Should be used only in cases when the reported coordinates do not correspond to real position of mechanics.

Find Move Range starts the initialization in the defined direction and then moves to the other limit and finds the whole range of the axes.

Parameter Input

Axis (required)

Bit 0 – Initialize X axis (default is 1)

Bit 1 – Initialize Y axis (default is 1)

Bit 2 – Initialize Z axis (default is 1)

Bit 3 – Initialize T axis (default is 1)

Direction

Bit 0 – Initialize X axis in plus direction; 0=X-, 1=X+ (default is 0)

Bit 1 – Initialize Y axis in plus direction; 0=Y-, 1=Y+ (default is 0)

Bit 2 – Initialize Z axis in plus direction; 0=Z-, 1=Z+ (default is 0)

Bit 3 – Initialize T axis in plus direction; 0=Z-, 1=T+ (default is 0)

Move Range

Bit 0 – Initialize X axis with move range scan; 1=X (default is 0)
Bit 1 – Initialize Y axis with move range scan; 1=Y (default is 0)
Bit 2 – Initialize Z axis with move range scan; 1=Z (default is 0)
Bit 3 – Initialize T axis with move range scan; 1=T (default is 0)

Example

```
InitChuck 7 0 0
```

MoveChuck

Description

Moves the chuck stage to the specified X/Y position. If chuck Z is in contact height or higher, Interlock and Auto Z flags will be analyzed and the stage will behave correspondingly; it can move to separation first or return an error.

Parameter Input

X Value (optional)

Y Value (optional)

PosRef (optional)

H – Home (default)
Z – Zero
C – Center
R – Current position

Unit (optional)

Y – Microns (default)
I – Mils
X – Index
J – Jog

Velocity (optional)

Velocity in percent (default is 100%)

Comp. level (optional)

D = Default – Use the default compensation. This is one of the four below (T, O, P, or N).
T = Technology – Use Prober, Offset, and Technology compensation (default).
O = Offset – Use Prober and Offset compensation.
P = Prober – Use only Prober compensation.
N = None – Does not use compensation.

Example

```
MoveChuck 5000 5000 R Y 100
```

MoveChuckAlign

Description

Moves the chuck Z axis to the align height. If no Contact height is set, returns an error.

Parameter Input

Velocity (optional)

Velocity in percent (default is 100%)

Example

```
MoveChuckAlign 70
```

MoveChuckContact

Description

Performs a movement of chuck Z axis to preset contact height or will return an error if no contact height is set.

Parameter Input

Velocity (optional)

Velocity in percent (default is 100%)

Example

```
MoveChuckContact 50
```

MoveChuckIndex

Description

Moves the chuck stage in index steps. This command modifies Die Home Position.

Parameter Input

X Index (required)

Index steps for X axis

Y Index (required)

Index steps for Y axis

PosRef (optional)

H – Home (default)
Z – Zero
C – Center
R – Current position

Velocity (optional)

Velocity in percent (default is 100%)

Example

```
MoveChuckIndex -1 0 R
```

MoveChuckLift

Description

Starts the chuck movement to the upper (0) or lower = lifted (1) position. The actual movement may take several seconds.

Parameter Input

EnableLift (optional)

0 – Upper position
1 – Down (lifted) position (default)

Example

```
MoveChuckLift
```

MoveChuckLoad

Description

Moves the Chuck stage in X,Y,Z and Theta to the load position.

Parameter Input

Load position (optional)

1 – Front Load position (default)

2 – Robot Load position

Example

```
MoveChuckLoad
```

MoveChuckSeparation

Description

Moves the chuck Z axis to the separation height. Returns error if no Contact height is set.

Parameter Input

Velocity (optional)

Velocity in percent (default is 100%)

Example

```
MoveChuckSeparation 70
```

MoveChuckSubsite

Description

Moves the chuck stage to the specified X,Y subsite position. Die home position is defined by destination position of last successful MoveChuck or MoveChuckIndex commands. MoveChuckVelocity does not touch die home position. If you have moved chuck with MoveChuckVelocity, “MoveChuckSubsite 0 0” will move it back to die home position.

Parameter Input

X Position (required)

Index steps for X axis

Y Position (required)

Index steps for Y axis

Unit (optional)

Y – Microns (default)

I – Mils

J – Jog

Velocity (optional)

Velocity in percent (default is 100%)

Example

```
MoveChuckSubsite 88.6 99.9 Y 100
```

MoveChuckTransfer

Description

Moves the Chuck to the Transfer Position.

If the movement starts in Load Position (where it's possible to pull out the chuck) the handling of the Add On Platen or the Pin Chuck is integrated.

Available for the following machine types: SUMMIT200, PA200PS, PA300PS, PA200PC, PA200PCE, PA300PC, PAV150, PAV200, PAV201, PA200PS-ATM, PA300PS-ATM, AP300PS-ATM, SPS300.

Example

```
MoveChuckTransfer
```

MoveChuckVelocity

Description

Moves the chuck stage in velocity mode. The motion continues until the StopChuckMovement command is received, or the end limit is reached.

Axes parameter:

'+' move this axis in plus direction

'-' move this axis in minus direction

'0' Do not change this axis

Parameter Input

X Axis (optional)

+, - or 0

Y Axis (optional)

+, - or 0

Z Axis (optional)

+, - or 0

X Velocity (optional)

Velocity in percent (default is 100%)

Y Velocity (optional)

Velocity in percent

Z Velocity *(optional)*

Velocity in percent

Example

```
MoveChuckVelocity + + 0 50.0 75.0 0
```

MoveChuckZ

Description

Moves the chuck Z axis to the specified height. If contact is set, only moves up to contact height will be allowed. If overtravel is enabled, chuck moves up to overtravel height.

Parameter Input

Z Value (optional)

PosRef (optional)

Z – Zero (default)
H – Contact
R – Current position

Unit (optional)

Y – Microns (default)
I – Mils
J – Jog

Z Velocity *(optional)*

Velocity in percent (default is 100%)

Comp. level (optional)

D = Default – Use the default compensation. This is one of the four below (T, O, P, or N).

T = Technology – Use Prober, Offset, and Technology compensation (default).
O = Offset – Use Prober and Offset compensation.
P = Prober – Use only Prober compensation.
N = None – Does not use compensation.

Example

```
MoveChuckZ 22344.7 Z Y 100
```

MoveChuckZSafe

Description

Moves the chuck Z-axis to a Z-height considered safe. This is either the lower Z-fence or the Kernel Setup command Chuck:SafeTransferHeight, whichever is larger.

Example

```
MoveChuckZSafe
```

ReadChuckHeights

Description

Returns the current settings used for the chuck Z movement.

Parameter Input

Unit (optional)

Y – Microns (default)
I – Mils

Parameter Output

Contact Height

Value of current contact height or -1 if contact height is not set.

Overtravel Gap

Value of overtravel gap added to contact height defines overtravel height.

Alignment Gap

Value of alignment gap subtracted from contact height defines alignment height.

Separation Gap

Value of separation gap subtracted from contact height defines separation height.

Search Gap

Value of search gap added to the contact height defines search height. It is the maximum move destination for auto contact search.

Example

```
ReadChuckHeights Y
10522.2 25.0 25.0 250.0 100.0
```

ReadChuckIndex

Description

The command gets the gets the current die size which is used by the software.

Parameter Input

Unit (optional)

Y – Microns (default)

I – Mils

Parameter Output

X Index

Y Index

Example

```
ReadChuckIndex Y
4600.0 4500.0
```

ReadChuckPosition

Description

Returns the actual chuck stage position in X,Y and Z. The default Compensation Mode is the currently activated compensation mode of the kernel.

Parameter Input

Unit (optional)

Y – Microns (default)

I – Mils

X – Index

PosRef (optional)

H – Home (default)

Z – Zero

C – Center

Comp. level (optional)

D = Default – Use the default compensation. This is one of the four below (T, O, P, or N).

T = Technology – Use Prober, Offset, and Technology compensation (default)

O = Offset – Use Prober and Offset compensation.

P = Prober – Use only Prober compensation.

N = None – Does not use compensation.

Parameter Output

X Value

Y Value

Z Value

Example

```
ReadChuckPosition Y Z
-1263.4 6766.7 -3336.7
```

ReadChuckStatus

Description

Returns the current chuck status. Every bit in the bit fields works like a boolean value: One <1> means true/on and zero <0> means false/off.

Parameter Output

IsInitialized

Bit 0 – X axis is initialized
Bit 1 – Y axis is initialized
Bit 2 – Z axis is initialized
Bit 3 – Theta axis is initialized

Mode

Bit 0 – Overtravel is enabled
Bit 1 – Auto Z is enabled
Bit 2 – Interlock is enabled
Bit 3 – Auto contact search
Bit 4 – Contact height is set
Bit 5 – Edge sensor interlock
Bit 6 – AutoQuiet is enabled
Bit 7 – Stage lock is enabled

OnEndLimit

Bit 0 – X on positive limit
Bit 1 – X on negative limit
Bit 2 – Y on positive limit
Bit 3 – Y on negative limit
Bit 4 – Z on positive limit
Bit 5 – X on negative limit
Bit 6 – Theta on positive limit
Bit 7 – Theta on negative limit

IsMoving

Bit 0 – X is moving
Bit 1 – Y is moving
Bit 2 – Z is moving
Bit 3 – Theta is moving

CompMode

T = Technology – Uses Prober, Offset, and Technology compensation
O = Offset – Uses Prober and Offset compensation
P = Prober – Uses only Prober compensation
N = None – Does not use compensation

Vacuum

0 – Turned off
1 – Turned on

Z Height

C – Contact height
S – Separation height
A – Alignment height
O – Overtravel height
T – Search height
0 – Undefined height

Load Position

Bit 0 – On load position one
Bit 1 – On load position two

Lift

0 – Up
1 – Down

ChuckCamera

0 – Undefined
C – On chuck
P – At park position

Example

```
ReadChuckStatus
7 7 0 0 T 1 C 2 0 P
```

ReadChuckThermoScale

Description

Read the linear scaling value of the chuck.

Parameter Output

Scale X

Linear scale factor

Scale Y

Linear scale factor

Example

```
ReadChuckThermoScale
0.99995 0.99994
```

ReadChuckThermoValue

Description

Read the actual set or calculated temperature.

Read the current temperature (either set or calculated) for the thermal scaling. With SetChuckThermoScale in use, the value will be calculated by the current scale and the expansion factor of silicium.

The temperature that is read with this command is not identical to the current chuck temperature.

Parameter Input

Unit (optional)

C – Celsius (default)
F – Fahrenheit

Parameter Output

Temperature
Linear scale factor

Expansion Coefficient X
*Exp-06

Expansion Coefficient Y
*Exp-06

Example

```
ReadChuckThermoValue C
240 2.33 2.43
```

SearchChuckContact

Description

Moves the chuck Z axis to the specified height and sets contact. The move stops immediately if the Contact Sensor (Edge Sensor) is triggered. If no contact has been found and an old one was set before the old contact will not get lost, it will be reset.

Parameter Input

Z Value (required)
Target Z position for the movement (default is 100)

PosRef (optional)
Z – Zero (default)
H – Contact
R – Current position

Unit (optional)
Y – Microns (default)
I – Mils
J – Jog

Z Velocity *(optional)*
Velocity in percent (default is 100%)

Comp. level (optional)
D = Default – Use the default compensation. This is one of the four below (T, O, P, or N).

T = Technology – Use Prober, Offset, and Technology compensation (default)
O = Offset – Use Prober and Offset compensation.
P = Prober – Use only Prober compensation.
N = None – Does not use compensation.

Parameter Output

Contact Height
Contact height from zero

Example

```
SearchChuckContact 1000 R Y 10
20010
```

SearchChuckContact requires additional hardware to be installed and configured.

Hardware • Mechanical switch (special probe tip) • Connected directly to the machine rear panel • Routed to the motion controller Z Axis - Axis In (input)
Function • In Contact or not connected • Open - no Z up move allowed
Can be enabled in ControlCenter.

SetChuckHeight

Description

Sets a predefined contact height and corresponding gaps for overtravel, alignment, and separation. If no data is selected for the optional parameters, contact height is set at the current position.

- If Mode is ‘0’, contact height can be set at current position. The levels O, A, S and T do not support Mode ‘0’.
- If Mode is ‘V’ and no value for height is specified, the default 0 will be used.
- If Mode is ‘R’, contact height can be invalidated. The levels O, A, S and T do not support Mode ‘R’.

Parameter Input

Level (optional)

C – Contact height (default)
O – Overtravel gap
A – Alignment gap
S – Separation gap
T – Search gap

Mode (optional)

0 – At current position (default)
R – Reset contact height
V – Take Value

Unit (optional)

Y – Microns (default)
I – Mils

Value (optional)

Example

```
SetChuckHeight C V Y 5000
```

SetChuckHome

Description

Sets wafer and die home position, which can be used later as coordinate system for movements.

Parameter Input

Mode (optional)

0 – Use current position (default)
V – Use given value

Unit (required if Mode is ‘V’)

Y – Microns (default)
I – Mils

X Value (required if Mode is 'V')

Y Value (required if Mode is 'V')

Example

```
SetChuckHome 0 Y
```

SetChuckIndex

Description

Sets the wafer index size.

Parameter Input

X Index (required)

Y Index (required)

Unit (optional)

Y – Microns (default)
I – Mils

Example

```
SetChuckIndex 2450.0 3200.0
```

SetChuckMode

Description

Mode manages the way the chuck behaves when it is in contact height. Chuck mode is made from 6 flags and you can control all of them using this command. Every flag can be turned on by using value 1 or turned off by using value 0. If you do not want to change a flag – use value of 2.

Parameter Input

Overtravel (optional)

Whenever the chuck should go to Contact height – it will move to Overtravel height instead.

0 – Off
1 – On
2 – (default)

Auto Z (optional)

If Chuck is in Contact height - move to the separation height before any X,Y movement.

0 – Off
1 – On
2 – (default)

Interlock (optional)

Disallows Chuck X, Y movement if the Z height is above the alignment height. This flag has effect only if Auto Z flag is off.

0 – Off
1 – On
2 – (default)

Edge sensor interlock (optional)

If activated: all X, Y and Theta movements are locked if the edge sensor is open.

0 – Off
1 – On
2 – (default)

Auto contact search (optional)

Set new contact height if the an edge sensor triggered.

0 – Off
1 – On
2 – (default)

Auto Quiet (optional)

After a move, the quiet mode is turned on automatically.

0 – Off
1 – On
2 – (default)

Example

```
SetChuckMode 0 1 1 2 0 2
```

SetChuckThermoScale

Description

Thermo drifts of the wafer will be compensated. The compensation is not persistent, so if the power is turned off, the compensation is deleted. The algorithm works as a backup.

- If you set a scaling factor, and then try to find this factor later, you must set the smaller, second result only. Iterative searching is supported in this case.
- If you set the temperature directly, the controller calculates the stage difference in the coordinate system.
- To reset the thermal scaling factors, use the command `SetChuckThermoValue 20`.

Parameter Input

Scale X (optional)

Linear scale factor (default is 1.0)

Scale Y (optional)

Linear scale factor (default is 1.0)

Example

```
SetChuckThermoScale 1.00005 1.000048
```

SetChuckThermoValue

Description

Set a temperature and optional the expansion coefficient. The temperature for normal level is 20°. The scaling factor at this temperature is 1.0. If you set the temperature directly the controller calculates the stage difference in the coordinate system. For the calculation use the set coefficient of expansion from the controller. The default coefficient of expansion is based on silicon: 2.33E-06 1/K.

Parameter Input

Temperature (required)

The default value is 20°C.

Unit (optional)

C – Celsius (default)
F – Fahrenheit

Coefficient of Expansion X (optional)

The default value is 0. If value is 0, then don’t change the coefficient

Coefficient of Expansion Y (optional)

The default value is 0. If value is 0, then use the X value

Example

```
SetChuckThermoValue 250 C 2.43 2.32
```

StopChuckMovement

Description

Stops chuck movement for the given axis.

Parameter Input

Axis (optional)

Bit 0 – 1 to stop X axis (default is 1)
Bit 1 – 1 to stop Y axis (default is 1)
Bit 2 – 1 to stop Z axis (default is 1)
Bit 3 – 1 to stop theta axis (default is 1)

Example

```
StopChuckMovement 7
```

Positioner Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands

InitProbe	MoveProbeLift	ReadProbeHeights	SetProbeHome
MoveProbe	MoveProbeSeparation	ReadProbeIndex	SetProbeIndex
MoveProbeAlign	MoveProbeVelocity	ReadProbePosition	SetProbeMode
MoveProbeContact	MoveProbeZ	ReadProbeStatus	StopProbeMovement
MoveProbeIndex	OrientProbe	SetProbeHeight	

InitProbe

Description

Performs an initialization move and resets the coordinate system of the given positioner.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Axis (required)

Bit 0 – Initialize X axis (default is 1)
Bit 1 – Initialize Y axis (default is 1)
Bit 2 – Initialize Z axis (default is 1)

Direction

Bit 0 – Initialize X axis in plus direction; 0=X-, 1=X+ (default is 0)
Bit 1 – Initialize Y axis in plus direction; 0=Y-, 1=Y+ (default is 0)
Bit 2 – Initialize Z axis in plus direction; 0=Z-, 1=Z+ (default is 0)

Move Range

Bit 0 – Initialize X axis with move range scan; 1=X (default is 0)
Bit 1 – Initialize Y axis with move range scan; 1=Y (default is 0)
Bit 2 – Initialize Z axis with move range scan; 1=Z (default is 0)

InitInPlace

Bit 0 – Initialize X with init in place
Bit 1 – Initialize Y with init in place
Bit 2 – Initialize Z with init in place

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
InitProbe 2 4 0 0 4
Initializes the Z axis of Positioner 2 with Init-In-Place
```

MoveProbe

Description

Moves a defined positioner to an X/Y position relative to the PosRef specified reference position.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

X Value (required)

Y Value (required)

PosRef (optional)

H – Home (default)
Z – Zero
C – Center
R – Current position

Unit (optional)

Y – Microns (default)
I – Mils
J – Jog
X – Index

Velocity (optional)

Velocity in percent (default is 100%)

Comp. level (optional)

D = Default – Use the default compensation. This is one of the four below (T, O, P, or N).

T = Technology – Use Prober, Offset, and Technology compensation (default).
O = Offset – Use Prober and Offset compensation.
P = Prober – Use only Prober compensation.
N = None – Does not use compensation.

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
MoveProbe 1 750 540 R Y 100
1
```

MoveProbeAlign

Description

Moves the given positioner Z axis to the alignment height.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Velocity (optional)

Velocity in percent (default is 100%)

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
MoveProbeAlign 1
1
```

MoveProbeContact

Description

Moves the given positioner Z axis to the preset contact height. If no contact height is set, the kernel will return a “Contact height not set” error.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Velocity (optional)

Velocity in percent (default is 100%)

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
MoveProbeContact 1 70
1
```

MoveProbeIndex

Description

Moves the selected positioner the given number of index steps.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

X Value (required)

Y Value (required)

PosRef (optional)

H – Home (default)
Z – Zero
C – Center
R – Current position

Velocity (optional)

Velocity in percent (default is 100%)

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
MoveProbeIndex 1 -2 -2 R 100
1
```

MoveProbeLift

Description

Moves the positioner 1, 2, 3 or 4 to the lower (0) or upper = lifted (1) position. The command initiates the motion. The whole movement may take some seconds.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

EnableLift (optional)

0 – Low position
1 – Lifted position (default)

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
MoveProbeLift 1 1
1
```

MoveProbeSeparation

Description

Moves the given positioner's Z axis to the separation height.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Velocity (optional)

Velocity in percent (default is 100%)

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
MoveProbeSeparation 1 70
1
```

MoveProbeVelocity

Description

Moves the specified positioner axis in velocity mode. The motion continues until the StopProbeMovement command is received, or the end limit is reached.

Axis parameter:

'+' move this axis in plus direction

'-' move this axis in minus direction

'0' Do not change this axis

Parameter Input
Probe no. (required) 1 – Positioner 1 2 – Positioner 2 3 – Positioner 3 4 – Positioner 4 5 – Positioner 5 6 – Positioner 6
X Axis (required) +, - or 0
Y Axis (required) +, - or 0
Z Axis (optional) +, - or 0
X Velocity (optional) Velocity in percent (default is 100%)
Y Velocity (optional) Velocity in percent (default is 0%)
Z Velocity *(optional)* Velocity in percent (default is 0%)

Parameter Output
Echo probe no. 1 – Positioner 1 2 – Positioner 2 3 – Positioner 3 4 – Positioner 4 5 – Positioner 5 6 – Positioner 6

Example

```
MoveProbeVelocity 1 + + 0 50 50 0
1
```

MoveProbeZ

Description

Moves a given positioner's Z axis to a defined Z height.

Parameter Input
Probe no. (required) 1 – Positioner 1 2 – Positioner 2 3 – Positioner 3 4 – Positioner 4 5 – Positioner 5 6 – Positioner 6
Z Value (optional)

PosRef (optional)

Z – Zero (default)
H – Focus height
R – Current position

Unit (optional)

Y – Microns (default)
I – Mils
J – Jog

Z Velocity *(optional)*

Velocity in percent (default is 100%)

Comp. level (optional)

D = Default – Use the default compensation. This is one of the four below (T, O, P, or N).

T = Technology – Use Prober, Offset, and Technology compensation (default)
O = Offset – Use Prober and Offset compensation.
P = Prober – Use only Prober compensation.
N = None – Does not use compensation.

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
MoveProbeZ 1 2234.7 Z Y 70
1
```

OrientProbe

Description

Defines the orientation of the positioner's coordinate system and turns the Y-axis of the probe coordinate system.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

EnableLift (optional)

R – Right side
L – Left side
0 – Left front position
1 – Left rear position
2 – Right front position
3 – Right rear position

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
OrientProbe 1 L
1
```

ReadProbeHeights

Description

Returns the current settings used for the chuck Z movement.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Unit (optional)

Y – Microns (default)
I – Mils

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Contact Height

Value of current contact height or -1 if contact height is not set.

Overtravel Gap

Value of overtravel gap added to contact height defines overtravel height.

Alignment Gap

Value of alignment gap subtracted from contact height defines alignment height.

Separation Gap

Value of separation gap subtracted from contact height defines separation height.

Search Gap

Value of search gap added to the contact height defines search height. It is the maximum move destination for auto contact search.

Example

```
ReadProbeHeights 1 Y
1 4000 50 25 250
```

ReadProbeIndex

Description

Returns the current positioner's wafer index values or the current positioner's index positions for X and Y.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Unit (optional)

Y – Microns (default)
I – Mils

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

X Value

Y Value

Example

```
ReadProbeIndex 1
1 460.0 450.0
```

ReadProbePosition

Description

Returns the actual positioner's position in X, Y and Z. The default Compensation Mode is the currently activated compensation mode of the kernel.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Unit (optional)

Y – Microns (default)
I – Mils
X – Index

PosRef (optional)

H – Home (default)
Z – Zero
C – Center

Comp. level (optional)

D = Default – Use the default compensation. This is one of the four below (T, O, P, or N).

T = Technology – Use Prober, Offset, and Technology compensation (default)
O = Offset – Use Prober and Offset compensation.
P = Prober – Use only Prober compensation.
N = None – Does not use compensation.

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

X Position

Y Position

Z Position

Example

```
ReadProbePosition 1
1 -1263.4 6766.7 -3336.7
```

ReadProbeStatus

Description

Returns the current positioner's status.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

IsInit

Bit 0 – X axis is initialized
Bit 1 – Y axis is initialized
Bit 2 – Z axis is initialized

Mode

Bit 0 – Overtravel is enabled
Bit 1 – Auto Z is enabled
Bit 2 – Interlock is enabled
Bit 3 – AutoZFollow enabled
Bit 4 – Contact height is set
Bit 5 – Reserved
Bit 6 – Reserved
Bit 7 – Stage lock is enabled

OnEndLimit

Bit 0 – X on positive limit
Bit 1 – X on negative limit
Bit 2 – Y on positive limit
Bit 3 – Y on negative limit
Bit 4 – Z on positive limit
Bit 5 – Z on negative limit

IsMoving

Bit 0 – X is moving
Bit 1 – Y is moving
Bit 2 – Z is moving

CompMode

T = Technology – Uses Prober, Offset, and Technology compensation
O = Offset – Uses Prober and Offset compensation
P = Prober – Uses only Prober compensation
N = None – Does not use compensation

Orientation

R – Right side
L – Left side
0 – Left front position
1 – Left rear position
2 – Right front position
3 – Right rear position

Z Height

C – Contact height
S – Separation height
A – Alignment height
O – Overtravel height
0 – Undefined height

Lift

0 – Down
1 – Up

Example

```
ReadProbeStatus 1
1 7 7 0 0 N R C 0
```

SetProbeHeight

Description

Defines the predefined contact height and corresponding gaps for overtravel, align, load and separation height. No data sets contact height at current position.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Level (optional)

C – Contact height (default)
O – Overtravel gap
A – Alignment gap
S – Separation gap

Mode (optional)

0 – At current position (default)
R – Reset contact height
V – Take Value

Unit (optional)

Y – Microns (default)
I – Mils

Value (optional)

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
SetProbeHeight 1 C 0
1
```

SetProbeHome

Description

Sets the positioner's Home position in X and Y. This position identifies the probe coordinate system for later movements. Usually this position is identical to the die home position.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Mode (optional)

0 – Use current position (default)
V – Use given value

Unit (required if Mode is 'V')

Y – Microns (default)
I – Mils

X Position (required if Mode is 'V')

Y Position (required if Mode is 'V')

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
SetProbeHome 1 0 Y
1
```

SetProbeIndex

Description

Sets the positioner's index size or the location of the reference die relative to the home die.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

X Index (required)

Y Index (required)

Unit (optional)

Y – Microns (default)
I – Mils

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
SetProbeIndex 1 245 320 Y
1
```

SetProbeMode

Description

The mode manages the way the chuck behaves when it is in contact height. Positioner mode is made up from 6 flags and the user can control all of them by using this command. Every flag can be turned on by setting value 1, or turned off by setting value 0. Use the value 2 if no change for a flag is needed.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Overtravel (optional)

0, 1 or 2 (default is 2)

Auto Z (optional)

0, 1 or 2 (default is 2)

Interlock (optional)

0, 1 or 2 (default is 2)

Auto Z Follow (optional)

0, 1 or 2 (default is 2)

Auto Quiet (optional)

0, 1 or 2 (default is 2)

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Overtravel

Positioner will move to the preset overtravel position.

Auto Z

Positioner will move to the separate position before all X, Y movements.

Interlock

Disallows positioner X, Y movement if the Z axis position is in contact or below the align position.

Auto Z Follow

Forces synchronization of Auto Z of the positioner and the chuck. Flags of ChuckContact, ProbeContact, ChuckAutoZ and ProbeAutoZ are required. The positioner moves in separation before chuck movement. The positioner moves in contact after chuck is in contact.

Auto Quiet

After a move, the quiet mode is automatically turned on if the chuck is in contact.

Example

```
SetProbeMode 1 0 1 1 0 2
1
```

StopProbeMovement

Description

Stops positioner movement immediately for the given axes. A smooth stop is performed.

Parameter Input

Probe no. (required)

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Axis (optional)

Bit 0 – Use X axis (default is 1)
Bit 1 – Use Y axis (default is 1)
Bit 2 – Use Z axis (default is 1)

Parameter Output

Echo probe no.

1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
StopProbeMovement 1
1
```

Scope Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands

InitScope	MoveScopeSeparation	ReadScopeIndex	SetScopeHeight
MoveScope	MoveScopeSilo	ReadScopePosition	SetScopeHome
MoveScopeAlign	MoveScopeVelocity	ReadScopeSilo	SetScopeIndex
MoveScopeFocus	MoveScopeZ	ReadScopeSiloCount	SetScopeMode
MoveScopeIndex	ReadScopeHeights	ReadScopeStatus	StopScopeMovement
MoveScopeLift			

InitScope

Description

Initializes the microscope stage in X, Y and Z. The Axis default is all axes, and the Direction default is XY in minus and Z in plus. Should be used only in cases when the reported coordinates do not correspond to real position of mechanics.

Find Move Range starts the initialization in the defined direction and then moves to the other limit and finds the whole range of the axes.

Parameter Input

Axis (required)

Bit 0 – Initialize X axis (default is 1)
Bit 1 – Initialize Y axis (default is 1)
Bit 2 – Initialize Z axis (default is 1)

Direction

Bit 0 – Initialize X axis in plus direction; 0=X-, 1=X+ (default is 0)
Bit 1 – Initialize Y axis in plus direction; 0=Y-, 1=Y+ (default is 0)
Bit 2 – Initialize Z axis in plus direction; 0=Z-, 1=Z+ (default is 0)

Move Range

Bit 0 – Initialize X axis with move range scan; 1=X (default is 0)
Bit 1 – Initialize Y axis with move range scan; 1=Y (default is 0)
Bit 2 – Initialize Z axis with move range scan; 1=Z (default is 0)

Example

```
InitScope 3 0 0
```

MoveScope

Description

Moves the microscope stage to the specified X/Y position relative to the PosRef specified reference position.

Parameter Input

X Value (optional)

Y Value (optional)

PosRef (optional)

H – Home (default)
Z – Zero
C – Center
R – Current position

Unit (optional)

Y – Microns (default)
I – Mils
J – Jog
X – Index

Velocity (optional)

Velocity in percent (default is 100%)

Comp. level (optional)

D = Default – Use the default compensation. This is one of the four below (T, O, P, or N).

T = Technology – Use Prober, Offset, and Technology compensation (default).
O = Offset – Use Prober and Offset compensation.
P = Prober – Use only Prober compensation.
N = None – Use no compensation.

Example

```
MoveScope 7500 5400 R
```

MoveScopeAlign

Description

Moves the Scope Z axis to the alignment height. If no focus height is set an error will be returned.

Parameter Input

Velocity (optional)

Velocity in percent (default is 100%)

Example

```
MoveScopeAlign 70
```

MoveScopeFocus

Description

Performs a move of the scope Z axis to preset focus height or return an error if no focus height is set.

Parameter Input

Velocity (optional)

Velocity in percent (default is 100%)

Example

```
MoveScopeFocus 50
```

MoveScopeIndex

Description

Moves the microscope stage in index steps. If no PositionReference byte is passed, the scope will step relative to the wafer home position. (‘R’ means the step relative to current position).

Parameter Input

X Index (required)

Index steps for X axis, default is 0

Y Index (required)

Index steps for Y axis, default is 0

PosRef (optional)

H – Home (default)
Z – Zero
C – Center
R – Current position

Velocity (optional)

Velocity in percent (default is 100%)

Example

```
MoveScopeIndex -1 0 R
```

MoveScopeLift

Description

Moves the microscope lift to the lower (0) or upper = lifted (1) position. This initiates the motion only, the actual movement may take some seconds.

Parameter Input

EnableLift (optional)

0 – Low position
1 – Lifted position (default)

Example

```
MoveScopeLift
```

MoveScopeSeparation

Description

Moves the scope Z axis to the separation height. Returns an error if no focus height is set.

Parameter Input

Velocity (optional)

Velocity in percent (default is 100%)

Example

```
MoveScopeSeparation 70
```

MoveScopeSilo

Description

Move the scope to the reference position of the given silo. The reference position should be - if not defined otherwise - 200 µm above the safe Z-height in the center of the scope.

Parameter Input

Silo

The number of the target silo.

Example

```
MoveScopeSilo 1
```

MoveScopeVelocity

Description

Moves the microscope stage in velocity mode. The motion continues until the StopScopeMovement command is received, or the end limit (error condition) is reached.

Axes parameter:

'+' move this axis in plus direction

'-' move this axis in minus direction

'0' Do not change this axis

Parameter Input

X Axis (optional)

+, - or 0 (default is Fixed)

Y Axis (optional)

+, - or 0 (default is Fixed)

Z Axis (optional)

+, - or 0 (default is Fixed)

X Velocity (optional)

Velocity in percent (default is 100%)

Y Velocity (optional)

Velocity in percent (default is 0)

Z Velocity *(optional)*

Velocity in percent (default is 0)

Example

```
MoveScopeVelocity + + 0 50 50 0
```

MoveScopeZ

Description

Moves the microscope Z axis to the specified height. Default velocity is 100%.

Parameter Input

Z Value (optional)

PosRef (optional)
Z – Zero (default)
H – Focus height
R – Current position

Unit (optional)
Y – Microns (default)
I – Mils
J – Jog

Z Velocity *(optional)*
Velocity in percent (default is 100%)

Comp. level (optional)
D = Default – Use the default compensation. This is one of the four below (T, O, P, or N).

T = Technology – Use Prober, Offset, and Technology compensation (default)
O = Offset – Use Prober and Offset compensation.
P = Prober – Use only Prober compensation.
N = None – Does not use compensation.

Example

```
MoveScopeZ 2344.7 R Y 75
```

ReadScopeHeights

Description

Returns the actual settings of the focus height, align gap, and separation gap for the scope Z axis.

Parameter Input

Unit (optional)
Y – Microns (default)
I – Mils

Parameter Output

Focus height

Alignment Gap

Separation Gap

Example

```
ReadScopeHeights Y
12000 25 250
```

ReadScopeIndex

Description

Returns the actual scope stage index values in X and Y.

Parameter Input

Unit (optional)
Y – Microns (default)
I – Mils

Parameter Output

X Index

Y Index

Example

```
ReadScopeIndex Y
4600.0 4500.0
```

ReadScopePosition

Description

Returns the actual scope stage position in X, Y and Z. The default Compensation Mode is the currently activated compensation mode of the kernel.

Parameter Input

Unit (optional)

Y – Microns (default)
I – Mils
X – Index

PosRef (optional)

H – Home (default)
Z – Zero
C – Center

Comp. level (optional)

D = Default – Use the default compensation. This is one of the four below (T, O, P, or N).

T = Technology – Use Prober, Offset, and Technology compensation (default)
O = Offset – Use Prober and Offset compensation.
P = Prober – Use only Prober compensation.
N = None – Does not use compensation.

Parameter Output

X Value

Y Value

Z Value

Example

```
ReadScopePosition Y Z
-1263.4 6766.7 3336.7
```

ReadScopeSilo

Description

Returns the definition of a silo. If the type is rectangle, center means _Point1_. If the type is circle, the meaning of Pos2X and Pos2Y is undefined.

Parameter Input

Position (required)

The number of the target silo. The default is 1.

Parameter Output

Type

CenterX

CenterY

Radius

Pos2X

Pos2Y

ZHigh

X
X-coordinate of the silo from zero in µm. The default is Compensated.

Y
Y-coordinate of the silo from zero in µm. The default is Compensated.

Z
Z-coordinate of the silo from zero in µm. The default is Compensated.

Radius
Radius of the silo in µm

Examples

```
ReadScopeSilo 1
120000 160000 110000 5000
```

ReadScopeSiloCount

Description

Returns the number of silos in the scope fence. Will be zero if no silos are configured.

Parameter Output

Count
Returns the number of silos, or returns 0 if there are no silos.

Example

```
ReadScopeSiloCount
5
```

ReadScopeStatus

Description

Returns the current scope status.

Parameter Output

IsInitialized
Bit 0 – X axis is initialized
Bit 1 – Y axis is initialized
Bit 2 – Z axis is initialized

OnEndLimit
Bit 0 – X on positive limit
Bit 1 – X on negative limit
Bit 2 – Y on positive limit
Bit 3 – Y on negative limit
Bit 4 – Z on positive limit
Bit 5 – Z on negative limit

IsMoving

Bit 0 – X is moving
Bit 1 – Y is moving
Bit 2 – Z is moving

CompMode

T = Technology – Uses Prober, Offset, and Technology compensation
O = Offset – Uses Prober and Offset compensation
P = Prober – Uses only Prober compensation
N = None – Does not use compensation

Lift

0 – Down
1 – Up
2 – Undefined

Z Height

S – Separation height
A – Alignment height
F – Focus height
0 – Undefined height

ScopeLight

0 – Off
1 – On

Mode

Bit 0 – Reserved
Bit 1 – ScopeFollowMode
Bit 2 – UseExternalFollowMode
Bit 3 – Reserved
Bit 4 – Focus is set
Bit 5 – Reserved
Bit 6 – AutoQuiet mode
Bit 7 – Stage lock is enabled

Example

```
ReadScopeStatus
7 0 0 T 0 S 1 16
```

SetScopeHeight

Description

This command defines scope focus height and the corresponding gaps for alignment or separation. No data sets focus height at current position.

Parameter Input

Level (optional)

F – Focus height (default)
A – Alignment gap
S – Separation gap

Mode (optional)

0 – At current position (default)

R – Reset contact height

V – Take Value

Unit (optional)

Y – Microns (default)

I – Mils

Value (optional)

Example

```
SetScopeHeight F V Y 5698.5
```

SetScopeHome

Description

Sets the scope Home position in X and Y. It identifies the scope coordinate system for later movements. Usually this position is identical to the die home location.

Parameter Input

Mode (optional)

0 – Use current position (default)

V – Use given value

Unit (required if Mode is 'V')

Y – Microns (default)

I – Mils

X Value (required if Mode is 'V')

Y Value (required if Mode is 'V')

Example

```
SetScopeHome 0
```

SetScopeIndex

Description

Sets the microscope wafer index.

Parameter Input

X Index (required)

Y Index (required)

Unit (optional)

Y – Microns (default)

I – Mils

Example

```
SetScopeIndex 5000. 5000. Y
```

SetScopeMode

Description

Set Scope mode contains a flag to set Auto Quiet mode and a flag to set Scope Follow mode. Each flag can be turned on by using value 1 or turned off by using value 0. If you do not want to change a flag, use value of 2.

- Auto Quiet – If enabled, quiet mode is automatically turned on after a move.
- ScopeFollowMode – If enabled, synchronizes the relative distance between the scope and the chuck after a move. At the same time, the current position of the motion channel for the chuck Z and the motion channel for the scope Z are initialized.

Parameter Input

AutoQuiet

0 – Disable
1 – Enable
2 – Do nothing (default)

ScopeFollowMode

0 – Disable
1 – Enable
2 – Do nothing (default)

Example

```
SetScopeMode 2 2
```

StopScopeMovement

Description

Stops scope movement for the given axis. A smooth stop is executed, no emergency stop.

Parameter Input

Axis (optional)

Bit 0 – X axis (default is 1)
Bit 1 – Y axis (default is 1)
Bit 2 – Z axis (default is 1)

Example

```
StopScopeMovement 7
```

AUX Site Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands				
CleanProbeTip	MoveAuxSite	ReadAuxStatus	SetAuxMode	SetAuxSiteType
GetAuxSiteCount	ReadAuxHeights	SetAuxHeight	SetAuxSiteCount	SetAuxThetaHome
GetAuxSiteName	ReadAuxIndex	SetAuxHome	SetAuxSiteName	SetCleaningParams
GetCleaningParams	ReadAuxPosition	SetAuxIndex		

CleanProbeTip

Description

Starts a probe tip cleaning on the given AUX site. The cleaning algorithm depends on the type of the AUX site and on the duration and the cleaning count that is set in configuration. The cleaning algorithm includes both a move to the pad site and a move back to the source site.

If the site defined by AUX ID is the wafer site, or is not a cleaning site, an error is returned. If no AUX ID is given, the first valid cleaning site is chosen.

Note: *in the case of a brush cleaning pad, XY moves in contact height are performed.*

Parameter Input

AUX ID (optional)

Parameter Output

AUX ID

The site where the cleaning was performed.

Example

```
CleanProbeTip
1
```

GetAuxSiteCount

Description

Reads the number of available AUX sites and the ID of the actual active AUX site. If this is 0, the chuck stage is currently active.

Parameter Output

Count

AUX ID

The site where the cleaning was performed.

Example

```
GetAuxSiteCount
2 0
```

GetAuxSiteName

Description

This command reads the description of an AUX site. If AUX ID is set to 0, the description of the chuck stage is given back. If no AUX ID is given, it tries to read from the active site.

Parameter Input

AUX ID (optional)

Parameter Output

AUX ID

The site from which the name was read.

Name

Example

```
GetAuxSiteName 1
1 Cal site (left)
```

GetCleaningParams

Description

Reads the cleaning parameters of a given AUX site. It is not possible to read cleaning parameters of the wafer site (AUX site 0). If no AUX ID is given, it tries to read the parameters from the active site.

Parameter Input

AUX ID (optional)

Parameter Output

AUX ID

The site from which the cleaning parameters are read.

Cleaning count

The cleaning count defines how many times the cleaning is repeated.

Cleaning time

The cleaning time defines how many milliseconds the needles are in the cleaning position in each cleaning cycle.

Remaining cleans

The number of clean operations that can be performed before the cleaning substrate is used up.

Example

```
GetCleaningParams 2
2 5 1000 233
```

MoveAuxSite

Description

Moves the chuck to the position of a given AUX site. A safe height is used for the move. If AUX ID is set to 0, the target of the move is the wafer site.

Parameter Input

AUX ID (optional)

Parameter Output

AUX ID

AUX ID is the new active site.

Example

```
MoveAuxSite 1
1
```

ReadAuxHeights

Description

Returns the actual technology heights from an AUX site. If AUX ID is set to 0, all response values are read from chuck stage. If no AUX ID is given, it tries to read the heights from the active site.

Parameter Input

AUX ID (optional)

Unit (optional)

Y – Microns (default)

I – Mils

Parameter Output

AUX ID

Contact Height

Value of current contact height or -1 if contact height is not set.

Overtravel Gap

Value of overtravel gap added to contact height defines overtravel height.

Alignment Gap

Value of alignment gap subtracted from contact height defines alignment height.

Separation Gap

Value of separation gap subtracted from contact height defines separation height.

Example

```
ReadAuxHeights 1 Y
1 4000 50 25 250
```

ReadAuxIndex

Description

Returns the actual AUX sites index values. If AUX ID is set to 0, the index size is read from chuck stage. If no AUX ID is given, it tries to read the index from the active site.

Parameter Input

AUX ID (optional)

Unit (optional)

Y – Microns (default)

I – Mils

Parameter Output

AUX ID (optional)

The site from which the index sizes were read.

X Index

Y Index

Example

```
ReadAuxIndex 1 Y
1 250.0 450.0
```

ReadAuxPosition

Description

Returns the actual AUX sites position in X, Y and Z. With AUX ID set to 0, the position is read for the chuck stage. If no AUX ID is given, it tries to read the position from the active site.

Parameter Input

AUX ID (optional)

Unit (optional)

Y – Microns (default)
I – Mils
X – Index

PosRef (optional)

H – Home (default)
Z – Zero
C – Center

Comp. level (optional)

D = Default – Use the default compensation. This is one of the four below (T, O, P, or N).

T = Technology – Use Prober, Offset, and Technology compensation (default)
O = Offset – Use Prober and Offset compensation.
P = Prober – Use only Prober compensation.
N = None – Does not use compensation.

Parameter Output

AUX ID (optional)

The site from which the position was read.

X Position

Y Position

Z Position

Example

```
ReadAuxPosition 1 Y Z
1 32500.0 15500.0 1235.8
```

ReadAuxStatus

Description

Returns the status of a single AUX site.

If AUX ID is set to 0, all response values are read from chuck stage. If no AUX ID is given, the values are read from the active AUX site.

Every bit in the bit fields works like a boolean value: One <1> means true/on and zero <0> means false/off.

Parameter Input

AUX ID (optional)

Parameter Output

AUX ID (optional)

The site from which the values were read.

Mode

Bit 0 – Overtravel is enabled
Bit 1 – Auto Z is enabled, read from chuck stage
Bit 2 – Interlock is enabled, read from chuck stage
Bit 3 – Auto contact search, read from chuck stage
Bit 4 – Contact height is set
Bit 5 – Edge sensor interlock, read from chuck stage
Bit 6 – AutoQuiet is enabled, read from chuck stage
Bit 7 – Stage lock is enabled, read from chuck stage

CompMode

T = Technology – Uses Prober, Offset, and Technology compensation
O = Offset – Uses Prober and Offset compensation
P = Prober – Uses only Prober compensation
N = None – Does not use compensation

Z Height

C – Contact height
S – Separation height
A – Alignment height
O – Overtravel height
T – Search height
0 – Undefined height

Aux Type

U – Not defined / Unknown
S – Calibration Substrate
C – Chuck video
P – Platen video
V – Contact verification
G – Pad cleaner
A – Air cleaner
B – Brush cleaner

Example

ReadAuxStatus 1

```
1 7 T C G
```

ResetCleaningPosition

Description

The ResetCleaningPosition command resets the current cleaning(switch) position back to the start position for the aux site. This allows the position to be reset when a new cleaning surface is installed.

Parameter Input

AuxSiteID (optional)

The ID of the aux site to reset. Defaults to current site.

XOffset (optional)

X offset from home to set the switch position. Defaults to 0.

YOffset (optional)

Y offset from home to set the switch position. Defaults to 0.

Parameter Output

AuxIDEcho

Example

```
ResetCleaningPosition 5
5
```

SetAuxHeight

Description

Sets the AUX sites contact height and corresponding gaps for overtravel, alignment and separation height. A contact height search gap for contact search with edge sensor can also be set. This search gap is always identical for all AUX sites. Without any optional parameters the command sets contact height to the current position.

If AUX ID is set to 0, the height is set for the chuck stage. If no AUX ID is given, contact height is set for the active site.

Parameter Input

AUX ID (optional)

Level (optional)

C – Contact height (default)
O – Overtravel gap
A – Alignment gap
S – Separation gap
T – Search gap

Mode (optional)

0 – At current position (default)
R – Reset contact height
V – Take Value

Unit (optional, if Mode = "V")

Y – Microns (default)
I – Mils

Value (optional, if Mode = "V")

Example

```
SetAuxHeight 1 C 0 Y 100
1
```

SetAuxHome

Description

Sets the AUX sites Home position in X and Y. It defines the origin of the AUX sites coordinate system for later movements. Usually this position is identical to the die home position.

If AUX ID is set to 0, home position is set for chuck stage. If no AUX ID is given, it tries to set the home position for the active site on current position.

Parameter Input

AUX ID (optional)

Mode (optional)

0 – Use current position (default)
V – Use given value

Unit (required if Mode is 'V')
Y – Microns (default)
I – Mils

X Position (required if Mode is 'V')

Y Position (required if Mode is 'V')

Parameter Output

AUX ID
The site for which the home position was set.

Example

```
SetAuxHome 1 0
1
```

SetAuxIndex

Description

Sets the AUX sites index size. If AUX ID is set to 0, index size is set for chuck stage.

Parameter Input

AUX ID (optional)

X Index (required)

Y Index (required)

Unit (optional)
Y – Microns (default)
I – Mils

Parameter Output

AUX ID
The site for which the index sizes were set.

Example

```
SetAuxIndex 1 245 320 Y
1
```

SetAuxMode

Description

Modes manage the way a stage behaves when it is in contact height. AUX site mode holds only a single flag. Flags can be turned on by using the value 1 or turned off by using the value 0. If you do not want to change the flag – use the value of 2. If AUX ID is set to 0, all flags are set for chuck stage. AUX site will move an additional overtravel on every contact move.

Parameter Input

AUX ID (optional)

Overtravel (optional)
0 – Deactivate overtravel
1 – Activate overtravel
2 – Leave as is (default)

Parameter Output

AUX ID
The site for which the flags were set.

Example

```
SetAuxMode 1
1
```

SetAuxSiteCount

Description

Sets the number of available AUX sites. For example, when preparing two cal chucks, the count must be set to 2. The system needs to be restarted for the changes to take effect.

Parameter Input

Count (required)

Example

```
SetAuxSiteCount 2
```

SetAuxSiteName

Description

Sets the description of an AUX site. With AUX ID zero, the description of the wafer site can be set.

Parameter Input

AUX ID (optional)

Name (optional)

Parameter Output

AUX ID

The site for which the name was set.

Example

```
SetAuxSiteName 1 Cal site (left)
1
```

SetAuxSiteType

Description

Sets the type for a given AUX site. A cleaning algorithm is available, depending on the type of AUX site. It is not possible to set the type of the wafer site (AUX site 0).

Parameter Input

AUX ID (optional)

Level

U – Not defined / Unknown
S – Calibration Substrate
C – Chuck video
P – Platen video
V – Contact verification
G – Pad cleaner
A – Air cleaner
B – Brush cleaner

Parameter Output

AUX ID

The site for which the type was set.

Example

```
SetAuxSiteType 2 G
2
```

SetAuxThetaHome

Description

Sets the AUX sites theta home position. It defines the origin of the AUX sites theta coordinate system for later movements. If AUX ID is set to 0, theta home position is set for chuck stage.

Parameter Input

AUX ID (optional)

Mode (optional)

0 – Use current position (default)

V – Use given value

Unit (required if Mode is 'V')

D – Degree (default)

A – Radian

Theta position (only used if Mode is 'V')

Default is 0.

Parameter Output

AUX ID

The site for which theta home was set.

Example

```
SetAuxThetaHome 1 0
1
```

SetCleaningParams

Description

Sets the cleaning parameters for a given AUX site. It is not possible to set the cleaning parameters of the wafer site (AUX site 0). If no AUX ID is given, it tries to set the parameters of the active site.

- The cleaning count defines how many times the cleaning is performed repeatedly.
- The cleaning time defines how many milliseconds the needles wait in cleaning position during each cleaning cycle.

Parameter Input

AUX ID (optional)

Cleaning count

Cleaning time

Parameter Output

AUX ID

The site for which the cleaning parameters were set.

Example

```
SetCleaningParams 2 5 1000
2
```

Theta Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands		
MoveTheta	ReadThetaPosition	SetThetaHome
MoveThetaVelocity	ReadThetaStatus	StopThetaMovement

MoveTheta

Description

Moves the chuck Theta axis to the specified position.

Parameter Input

Value (required)

PosRef (optional)

H – Home (default)
Z – Zero
C – Center
R – Current position

Unit (optional)

D – Degree (default)
A – Radian
J – Jog

Velocity (optional)

Velocity in percent (default is 100%)

Example

```
MoveTheta 3.87 Z D
```

MoveThetaVelocity

Description

Moves the chuck Theta axis in velocity mode. The motion continues until the StopThetaMovement command is received, or the end limit (error condition) is reached. Move CCW with 50% of the default axis speed. This command is used for joystick movements that do not have a target destination.

Parameter Input

PosRef (required)

0 – Move Theta in plus direction (CCW)
1 – Move Theta in minus direction (CW)

Velocity (optional)

Velocity in percent (default is 100%)

Example

```
MoveThetaVelocity + 50.0
```

ReadThetaPosition

Description

Returns the actual Theta position.

Parameter Input

Unit (optional)

D – Degree (default)
A – Radian

PosRef (optional)

H – Home (default)
Z – Zero
C – Center

Parameter Output

Theta Position

Example

```
ReadThetaPosition D Z
3.42
```

ReadThetaStatus

Description

Returns the actual status of the chuck Theta axis.

Parameter Output

IsInit

OnEndLimit

0 – Theta not in any end limit
1 – Theta in positive end limit
2 – Theta in negative end limit

IsMoving

Example

```
ReadThetaStatus
1 0 0
```

SetThetaHome

Description

Sets the chuck Theta position to home. This position is usable as reference position for other commands.

Parameter Input

Mode (optional)

0 – Use current position (default)
V – Use given value

Unit (required if Mode is 'V')

D – Degree (default)

A – Radian

Theta Position (only used if Mode is 'V')

Example

```
SetThetaHome 0
```

StopThetaMovement

Description

Stops any type (velocity, position etc.) of chuck Theta axis movement. This is treated as a smooth stop rather than an emergency stop.

Example

```
StopThetaMovement
```

Turret Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands	
ReadCurrentLens	SelectLens

ReadCurrentLens

Description

Returns the number of the current microscope lens. Because no motorized turret is supported by kernel, the command works in simulation mode.

Parameter Output

Lens

Lens no.

Example

```
ReadCurrentLens
4
```

SelectLens

Description

Rotates the microscope turret to the specified lens. Parfocality and para-centricity are adjusted according to the stored values. Because no motorized turret is supported by kernel, the command works in simulation mode.

Parameter Input

Lens (optional)

Lens no. (default is 1)

Example

```
SelectLens 3
```

System Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands			
EchoData	InkDevice	ReadWaferStatus	SetLoaderGate
EnableMotorQuiet	ReadContactCount	ReportKernelVersion	SetMicroLight
GetAxisReverse	ReadProberStatus	ResetContactCount	SetOutput
GetBackSideMode	ReadProbeSetup	SetBeaconStatus	SetSoftwareStop
GetDarkMode	ReadSensor	SetChuckVacuum	SetStageLock
GetSoftwareStop	ReadTypedSensor	SetDarkMode	SetTypedOutput
GetStageLock	ReadSystemStatus	SetExternalMode	StopAllMovements

EchoData

Description

Test Command for the Kernel Communication. Like a ping command, the given text string is returned unchanged.

Parameter Input

Test String (optional)

Parameter Output

Echo String

Example

```
EchoData Test
Test
```

EnableMotorQuiet

Description

Toggles the motor power for quiet probing. In most applications this feature is not required. However, for special measurements it may be necessary to reduce the noise of the system.

Quiet On turns off the motor power for all stages (or the one specified with the optional parameter).

Note: Turning off all motor power can cause small mechanical shifts in the micron range because of mechanical relaxation.

Quiet Off turns on the motor power of all stages (or the one specified with the optional parameter) for subsequent movements.

A move command will also turn on the motor power of the moved stage automatically. For automatic switching of the quiet mode refer to Auto Quiet functionality.

Parameter Input

Mode

0 – QuietMode Off, motor power On (default)

1 – QuietMode On, motor power Off

Stage (optional)
Stage for which the quiet mode should be activated or deactivated. If this parameter is not sent, quiet mode will be activated or deactivated for all stages.

Examples

Enables the quiet mode for all stages.
EnableMotorQuiet 1

Enables the quiet mode only for the chuck stage.
EnableMotorQuiet 1 C

Enables the quiet mode only for the Positioner1 stage.
`EnableMotorQuiet 1 1`

Disables the quiet mode only for the scope stage.
`EnableMotorQuiet 0 S`

GetAxisReverse

Description
Returns 1 if the positive and negative directions of an axis have been reversed.

Parameter Input

Stage
- C – Chuck (default)
- P – Platen
- S – Scope
- 1 – Positioner 1
- 2 – Positioner 2
- 3 – Positioner 3
- 4 – Positioner 4
- 5 – Positioner 5
- 6 – Positioner 6

Axis
- X – X Axis
- Y – Y Axis
- Z – Z Axis
- T – ThetaAxis

Parameter Output

Is reverse

Example

```
GetAxisReverse C X
1
```

GetBackSideMode

Description
Returns the current side mode.

Parameter Input

IsSet

0 – Topside mode

1 – Bottomside mode

Example

```
GetBackSideMode
1
```

GetDarkMode

Description

Returns the current mode of the light sources.

Parameter Output

State

0 – Bright

1 – Dark

Example

```
GetDarkMode
1
```

GetSoftwareStop

Description

Returns the current state of Stop mode.

Parameter Output

StopState

0 – Stop mode is inactive

1 – Stop mode is active

Example

```
GetSoftwareStop
1
```

GetStageLock

Description

Reads whether stages or joystick controller are locked.

The name of the application, which previously locked some stage, is given back in the Application Name parameter.

Parameter Input

Stage (required)

C – Chuck (default)
J – Joystick
P – Platen
S – Scope
1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Parameter Output

Status

0 – Lock off
1 – Lock on

Application name

Example

```
GetStageLock C
1 WaferMap
```

InkDevice

Description

Inks the current device under test. The range of PulseWidth is 20ms to 2000ms. For correct function, the inkers have to be linked to default outputs.

Parameter Input

Inker

Bit 0 – Inker 1
Bit 1 – Inker 2
Bit 2 – Inker 3
Bit 3 – Inker 4

PulseWidth (optional)

Pulse width in ms (default is 50ms)

Example

```
InkDevice 3 40
```

ReadContactCount

Description

Returns the number of times this stage moved to contact since the last time the counter was reset.

Parameter Input

Stage (required)

C – Chuck (default)
P – Platen
1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Parameter Output

Contact count

Number of contacts for this stage since last reset.

Example

```
ReadContactCount C
54682
```

ReadProberStatus

Description

Returns the current prober status.

Parameter Output

Busy

Bit 0 – Chuck
Bit 1 – Scope
Bit 2 – Theta
Bit 3 – Platen
Bit 4 – Positioner 1
Bit 5 – Positioner 2
Bit 6 – Positioner 3
Bit 7 – Positioner 4
Bit 8 – Positioner 5
Bit 9 – Positioner 6

InContact

Bit 0 – Chuck
Bit 1 – Platen
Bit 2 – Positioner 1
Bit 3 – Positioner 2
Bit 4 – Positioner 3
Bit 5 – Positioner 4
Bit 6 – Positioner 5
Bit 7 – Positioner 6

Mode

L – Local mode
R – Remote mode

Prober Quiet

0 – Not quiet
1 – Quiet

Example

```
ReadProberStatus
0 1 R 1
```

ReadProbeSetup

Description

Returns the current positioner configuration.

Parameter Output

Probe 1 Type
- 0 – No positioner
- 5 – PH 500
- 6 – PH 600

Probe 2 Type
- 0 – No positioner
- 5 – PH 500
- 6 – PH 600

Probe 3 Type
- 0 – No positioner
- 5 – PH 500
- 6 – PH 600

Probe 4 Type
- 0 – No positioner
- 5 – PH 500
- 6 – PH 600

Probe 5 Type
- 0 – No positioner
- 5 – PH 500
- 6 – PH 600

Probe 1 Axes
- Bit 0 – Has XY axes
- Bit 1 – Has Z axis

Probe 2 Axes
- Bit 0 – Has XY axes
- Bit 1 – Has Z axis

Probe 3 Axes
- Bit 0 – Has XY axes
- Bit 1 – Has Z axis

Probe 4 Axes
- Bit 0 – Has XY axes
- Bit 1 – Has Z axis

Probe 5 Axes
- Bit 0 – Has XY axes
- Bit 1 – Has Z axis

Probe 6 Type

0 – No positioner

5 – PH 500

6 – PH 600

Probe 6 Axes

Bit 0 – Has XY axes

Bit 1 – Has Z axis

Example

```
ReadProbeSetup
6 6 0 6 0 3 3 0 3 0
```

ReadSensor

Description

Returns the actual status of the specified input channel, output channel or edge sensor. The signal table is described in the Hardware Manual. Each used IO-board has 16 channels (max. 4 IO-boards are supported), what results in a domain of 64 channel numbers. Beyond this range there are pseudo channels, which are used for special purposes. Pseudo channels:

65 – Edge sensor one.

66 – Edge sensor two

Parameter Input

Channel (required)

Input channel number

Sensor Type (optional)

I – Input sensor (default)

O – Output sensor

E – Edge sensor

Parameter Output

Status

0 – Off

1 – On

Example

```
ReadSensor 3 I
1
```

ReadSystemStatus

Description

Returns the system options which are currently enabled at the prober. The Machine Type String is 16 byte fixed size.

Parameter Output

Machine type character and string

0 – MMC (Demo Mode)
1 – PM8
2 – PA200
3 – PA300
4 – PA200PS
5 – PA300PS
6 – PA300DSP
7 – PA200DSP
8 – PA200PC
9 – PA300PC
E – PAV150
F – PAV200
G – PAV201
H – Summit200
I – PA200PS-ATM
J – PA300PS-ATM
L – AP300PS-ATM
M – SPS300 (CM300)
N – PA200PCE (APS200Tesla)
O – NUCLEUS (Elite300, Summit)

Chuck XY

Has XY axis controller

Chuck Z

Has Z axis controller

Chuck Theta

Has Theta axis controller

Scope XY

Has XY axis controller

Scope Z

Bit 0 = Has Z axis controller
Bit 1 = 0 – Pneumatics; 1 – Motorized

EdgeSensor

Bit 0 = Has an edge sensor one
Bit 1 = Has an edge sensor two

UserMode

P – Protected mode
U – Unprotected mode

Turret

0 (false)

TemperatureChuck

Has a connected temperature chuck

AUX sites
Number of set AUX sites

Platen XY
Has XY axis controller

Platen Z
Has Z axis controller

LoaderGateState
State of the loader gate

Legacy system (probe station) type
0 – None
1 – Virtual12K
2 – VirtualS300
3 – Virtual12KTheta
4 – 12K
5 – S300
6 – E300
7 – E300Edge
8 – 12KTheta
9 – S300Theta

Example

```
ReadSystemStatus
PA200 2 1 1 1 1 3 2 P 0 0 2 0 0 U
```

ReadTypedSensor

Description

Returns the status of the specified input channel, output channel, or edge sensor.

Parameter Input

Port type
See Inputs and Outputs on page 221. Example: Read if vacuum is turned on.

Status (optional)
0 – Off (default)
1 – On

Example

```
ReadTypedSensor DO_WaferVacuum
1
```

Inputs and Outputs

Inputs	Outputs	
DI_Cassette1	DO_AirCleaner	DO_LoaderDoorLock
DI_Cassette2	DO_AirVacSwitch	DO_LoaderGateOpen
DI_ChuckLock	DO_AuxPower	DO_LoaderGateClose
DI_DarkMode	DO_AuxVacuum1	DO_LogoPurge1
DI_EdgeSensor1	DO_AuxVacuum2	DO_LogoPurge2
DI_EdgeSensor2	DO_BeaconRed	DO_PeltierCooler
DI_EMO	DO_BeaconYellow	DO_PinVacuum
DI_EPin	DO_BeaconGreen	DO_PlatenLift
DI_GraniteValid	DO_BeaconBlue	DO_ProbeCardPurge
DI_LoaderDoor	DO_BeaconWhite	DO_QuickPurge
DI_LoaderDoorButton	DO_BernoulliAddOn	DO_QuietLed
DI_LoaderGateClosed	DO_BrakeChuckZ	DO_ReadyLamp
DI_LoaderGateOpen	DO_BrakeScopeZ	DO_RollOutLock
DI_MotorPower	DO_Buzzer1	DO_ScopeValve
DI_PlatenPosUp	DO_Buzzer2	DO_ScopeLight
DI_PlatenPosDown	DO_BypassPurge	DO_ThetaBearing
DI_PressureGlobal	DO_CameraCool	DO_VacuumLED
DI_RollOutLockOn	DO_CameraQuiet	DO_VacuumRing1
DI_RollOutLockOff	DO_ChamberPurge	DO_VacuumRing2
DI_ScopeLock	DO_ChuckVacuum	DO_VacuumRing3
DI_ScopeDown	DO_ContactViewPurge	DO_VacuumRing4
DI_StationRestFront	DO_Discharge	DO_VibrationDamper
DI_StationRestRear	DO_EdgeSensor	DO_WaferVacuum
DI_VacuumButton	DO_FirstLoadLock	
DI_VacuumGlobal	DO_GPIBSwitchBox	
DI_WaferVacuumSensor2	DO_Inker1	
DI_WaferVacuumSensor3	DO_Inker2	
DI_WaferVacuumSensor	DO_Inker3	
	DO_Inker4	

ReadWaferStatus

Description

Returns whether the system detected a wafer. This feature can be used if the probe station is equipped with a vacuum sensor and while vacuum is activated.

Parameter Output

Wafer sensed

U – Unknown

C – Wafer connected (sensed)

D – Wafer disconnected

Example

```
ReadWaferStatus
C
```

ReportKernelVersion

Description

Returns the version information of the kernel or the control box code. The “Version” value contains version number and revision level of the actual Kernel implementation. The text string contains a code description, version number, and the revision date.

Parameter Input

Module (optional)

K – Kernel Code (default)
C – Control Box Code
O – Operating System
X – Control Box OS

Parameter Output

Version

Version String

Example

```
ReportKernelVersion K
4.01 “PA Kernel v4.01 built on Jun 30 2009”
```

ResetContactCount

Description

Resets the contact counter for the specified stage to zero.

Parameter Output

Stage

C – Chuck
P – Platen
1 – Positioner 1
2 – Positioner 2
3 – Positioner 3
4 – Positioner 4
5 – Positioner 5
6 – Positioner 6

Example

```
ResetContactCount C
```

SetBeaconStatus

Description

This command controls the beacon channel of the peripheral output board. All lights can be switched on, off, or blinking. Setting an interval to zero means, that the corresponding light will be statically on or off. The minimum blinking interval is 50ms.

Parameter Input

Mode (optional)

Bit 0 – Red light
Bit 1 – Green light
Bit 2 – Yellow light
Bit 3 – Blue light
Bit 4 – White light

Interval red

Pulse width in ms for red light (default is 1000ms)

Interval green

Pulse width in ms for green light (default is 0)

Interval yellow

Pulse width in ms for yellow light (default is 0)

Interval blue

Pulse width in ms for blue light (default is 0)

Interval white

Pulse width in ms for white light (default is 0)

Example

```
SetBeaconStatus 1 500 0 0
```

SetChuckVacuum

Description

Toggles the chuck vacuum on or off. Returns vacuum timeout error for stations with wafer vacuum sensors when no wafer is detected.

Parameter Input

Mode (optional)

0 – Turned off (default)
1 – Turned on

Example

```
SetChuckVacuum 1
```

SetDarkMode

Description

Switches off all LEDs on the Control Box, on the positioners, and on the screen of the Control Box.

Parameter Input

State (optional)

0 – Bright (default)
1 – Dark

Example

```
SetDarkMode 1
```

SetExternalMode

Description

If ‘R’ is sent, it sets the external mode of the kernel. This disables most of the control box functions (except the LOCAL function key). Otherwise, this command re-enables all functions of the control box.

Parameter Input

Mode (optional)

R – Remote mode (default)

L – Local mode

Example

```
SetExternalMode R
```

SetLoaderGate

Description

Opens or closes the loader gate. Automatic handling systems can load wafers to the chuck through the gate.

Parameter Input

Open (optional)

0 – Close gate (default)

1 – Open gate

Example

```
SetLoaderGate 1
```

SetMicroLight

Description

Toggles the microscope light channel of the peripheral output board and all cameras.

Parameter Input

Mode (optional)

0 – Illuminator off

1 – Illuminator on

2 – Toggle illuminator (default)

Example

```
SetMicroLight 1
```

SetOutput

Description

Controls the Velox output channel signals. It can be used to activate/deactivate outputs.

Parameter Input

Channel (required)

Channel number (default is 1)

Status (optional)

0 – Off (default)

1 – On

Pulse time

(Default is -1)

Example

```
SetOutput 4 0
```

SetSoftwareStop

Description

Activates or deactivates Stop mode.

Parameter Output

StopState

0 – Deactivate Stop mode

1 – Activate Stop mode

Example

```
SetSoftwareStop
1
```

SetStageLock

Description

Enables or disables the stage lock functionality. Stages which are locked can't be moved in any way. If the joystick controller is locked, the display and the keys are deactivated. The name of the application can be stored by using the Application Name parameter and is shown in lock-caused error messages.

Parameter Input

Stage (required)

C – Chuck

J – Joystick

P – Platen

S – Scope

1 – Positioner 1

2 – Positioner 2

3 – Positioner 3

4 – Positioner 4

5 – Positioner 5

6 – Positioner 6

Status (required)

0 – Lock off (default)

1 – Lock on

Application name (optional)

Example

```
SetStageLock J 1 WaferMap
```

SetTypedOutput

Description

Controls the kernel valve driver signals and can be used to drive the outputs.

Parameter Input

Output type

See Inputs and Outputs on page 221. Example: Deactivate the wafer vacuum.

Status (optional)

0 – Off (default)

1 – On

Pulse time integer

-1 is the default pulse time for the output channel (beacon only)

Example

```
SetTypedOutput DO_WaferVacuum
```

StopAllMovements

Description

Stops all Prober movements immediately. The response status value of any pending movement will signify that the stop command was executed.

Example

```
StopAllMovements
```

Offset Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands		
EnableOffset	GetOffsetInfo	SetOffset

EnableOffset

Description

Enables or disables the Offset XY compensation for the selected stage. If the compensation holds an offset different from zero, the chuck is automatically moved the distance of the offset. The chuck also moves automatically to a safe height.

The move can be disabled by the “Enable” parameter. This may put the stage outside the software fence.

Parameter Input

Stage (required)

C – Chuck (default)

S – Scope

P – Positioner

Enable (required)

0 – Disable

1 – Enable (default)

Move (optional)

0 – tbd

1 – xxx (default)

Example

```
EnableOffset C 1
```

GetOffsetInfo

Description

Gets information about the Offset XY compensation of the selected stage, including the stored offset values and whether the compensation is enabled or disabled.

Parameter Input

Stage (required)

C – Chuck (default)

Parameter Output

Enabled

X Value

Y Value

Example

```
GetOffsetInfo C
1 100000.0 0.0
```

SetOffset

Description

Sets the offset values for the Offset XY compensation for the selected stage. The changes take effect immediately. Note that the Offset XY compensation must be enabled for the values to take effect. The compensation is enabled or disabled using EnableOffset command.

Parameter Input

Stage (required)

C – Chuck (default)

X Offset (required)

Y Offset (required)

Example

```
SetOffset C 100000.0 0.0
```

Setup Commands

EnableEdgeSensor

Description

Enables/disables the use of an edge sensor.

Parameter Input

Sensor number

Number of the sensor to enable/disable

Enable sensor

Example

```
EnableEdgeSensor 1 1
```

Temperature Chuck Commands

The details for individual commands are provided. Descriptions, parameters, responses, and examples are included, as appropriate to each command.

Commands		
EnableHeaterHoldMode	GetHeaterTemp	SetHeaterSoak
EnableHeaterStandby	GetTargetTemp	SetHeaterTemp
GetDewPointTemp	HeatChuck	StopHeatChuck
GetHeaterSoak	ReadTemperatureChuckStatus	

EnableHeaterHoldMode

Description

Switches the Temperature Chuck devices hold mode on or off. In hold mode the chuck is heated with a constant current to avoid noise from the temperature control. The hold mode can be enabled only in HoldReady state.

Parameter Input

Mode (required)

0 – Temperature control on

1 – Temperature control off (default)

Parameter Output

Mode

0 – Temperature control on

1 – Temperature control off

Example

```
EnableHeaterHoldMode 1
1
```

EnableHeaterStandby

Description

Switches the power save (standby) mode of the device on or off. If the device is in power save mode, setting and reading temperatures switches off the power save mode automatically.

Parameter Input

Mode (required)

0 – Power save (standby) off

1 – Power save (standby) on (default)

Parameter Output

Mode

0 – Power save (standby) off

1 – Power save (standby) on

Example

```
EnableHeaterStandby 1
1
```

GetDewPointTemp

Description

Returns the current dew point temperature, if a dew point sensor is connected.

Parameter Input

Unit (optional)

C – Celsius (default)
F – Fahrenheit
K – Kelvin

Parameter Output

Temperature

Unit

C – Celsius
F – Fahrenheit
K – Kelvin

Example

```
GetDewPointTemp C
1.6 C
```

GetHeaterSoak

Description

Returns the soaking status and current soak time values in seconds.

Parameter Output

SoakTime

Fixed soak time, in seconds

Status

Status of fixed soak time

0 – Disabled
1 – Enabled

DynamicSoakTime

Dynamic soak time, in seconds (dependent on temperature delta between start and target temperature)

DynamicStatus

Status of dynamic soak time

0 – Disabled
1 – Enabled

FixedDieSoakTime

Fixed Die soak time, in seconds (dependent on temperature delta from ambient)

DynamicDieSoakTime

Dynamic Die soak time, in seconds (dependent on temperature delta from ambient)

FixedDieSoakStatus

Status of fixed die soak time

0 – Disabled
1 – Enabled

DynamicDieSoakStatus
Status of dynamic die soak time
0 – Disabled
1 – Enabled

Example

```
GetHeaterSoak
600 1 63.0 1 900.0 30.0 1 1
```

GetHeaterTemp

Description

Reads the current temperature of the chuck and determines the status of the thermal system.

Parameter Input

Unit (optional)
C – Celsius (default)
F – Fahrenheit
K – Kelvin

Parameter Output

Actual Temperature

Unit
C – Celsius
F – Fahrenheit
K – Kelvin

Status
0 – Unknown
1 – Heating
2 – Cooling
4 – At temperature
8 – Hold ready
16 – Holding
32 – Standby
64 – Soaking

Example

```
GetHeaterTemp
25.2 C 1
```

GetTargetTemp

Description

Reads the target temperature of the chuck.

Parameter Input

Unit (optional)
C – Celsius (default)
F – Fahrenheit
K – Kelvin

Parameter Output

Target Temperature

Unit

C – Celsius
F – Fahrenheit
K – Kelvin

Example

```
GetTargetTemp F
92.7 F
```

HeatChuck

Description

Sets a new target temperature and starts the heating or cooling of the chuck. A response to the command will be returned after reaching the given temperature and waiting the soak time or an unexpected interrupt of the process.

Parameter Input

New temperature (required)

Default is 25°C

Unit (optional)

C – Celsius (default)
F – Fahrenheit
K – Kelvin

ReduceContact (optional)

Default is 1

Parameter Output

Actual Temperature

Default is 0

Unit

C – Celsius
F – Fahrenheit
K – Kelvin

Example

```
HeatChuck 60.0 C
59.8 C
```

ReadTemperatureChuckStatus

Description

Returns values of the temperature chuck status. The status byte gives information about the current controller's action. The dew point sensor status encapsulates information, if such a sensor is connected, and if the actual dew point difference temperature is readable (active). It can be used for purge control. Soak time left (in seconds) is used when soak time is actually running.

Parameter Output

Status

0 – Unknown (default)
1 – Heating
2 – Cooling
4 – At temperature
8 – Hold ready
16 – Holding
32 – Standby
64 – Soaking

Dew point sensor status

Bit 0 – Dew point sensor connected
Bit 1 – Dew point sensor active

Soak time left

Returns the left soak time in seconds. Default is 0

EcoMode

Read if ECO mode is available (depending on thermal chuck and forced ECO mode state).

Example

```
ReadTemperatureChuckStatus
64 3 180
```

SetHeaterSoak

Description

Sets a new soak time value. The unit of the value is seconds. If soak time is actually running, it may be affected by the change.

Note: There is a difference between -1 and 0. If you set one of those to 0, the soak time will not be used. If you specify -1, the soak time will not be changed.

– Sending '`SetHeaterSoak -1 0 0 0`' will leave the fixed soak time unchanged set all other soak times to 0 seconds.
– Sending '`SetHeaterSoak 0 60 60 60`' will set the fixed soak time to 0 seconds and set all other soak times to 60s.

Parameter Input

Soak Time (default is 60)

Fixed soak time, in seconds, that the system waits after each temperature transition.

DynamicSoakTime (default is -1)

The amount of time, in seconds, that the system waits after each temperature transition:
dynamic soak time multiplied by the temperature difference to the last temperature.

FixedDieSoakTime (default is -1)

Fixed Die soak time, in seconds, that the system waits after moving to the next die.

DynamicDieSoakTime (default is -1)

The amount of time, in seconds, that the system waits after moving to the next die:
dynamic die soak time multiplied by the temperature difference to ambient

Example

```
SetHeaterSoak 600 63 900 30
0
```

SetHeaterTemp

Description

Sets a new target temperature and starts the heating or cooling of the temperature chuck. The new target temperature is returned immediately.

Parameter Input

New temperature (required)

Default is 25°C

Unit (optional)

C – Celsius (default)
F – Fahrenheit
K – Kelvin

UseContactSafely

Default is 1

Parameter Output

Target Temperature

Unit

C – Celsius
F – Fahrenheit
K – Kelvin

Example

```
SetHeaterTemp 60.0 C
60.0 C
```

StopHeatChuck

Description

Stops a pending heating or cooling process. If the device is not at temperature, the actual temperature is set as target temperature.

Example

```
StopHeatChuck
```

Notifications

SoftwareStopChangedNotify

Description

This notification is sent when the software Stop mode is activated or deactivated.

Parameter Output

SoftwareStopState

0 – Stop mode is inactive
1 – Stop mode is active

Example

```
SoftwareStopChangedNotify
1
```

AlertNotification

Description

This is not a regular command. Asynchronous notifications will be sent from the Kernel controller to the host system only.

Parameter Input

Notif. ID

Data 1

Data 2

Data 3

Data1, Data2, Data3 are integers. Characters must be interpreted as their ASCII value.

No.	Description	Data 1	Data 2	Data 3	Comment
01	Prober Reset	0	0	0	Sent on <Power on> or after reset of the controller
02	Extern Mode	L or R	0	0	Sent if prober is set in remote mode from command or panel
03	Stage lock	C – Chuck P – Platen S – Scope 1 – Positioner 1 2 – Positioner 2 3 – Positioner 3 4 – Positioner 4 5 – Positioner 5 6 – Positioner 6 J – Joystick controller	0 or 1	0	Sent if stage lock for a special stage or the joystick controller is activated or deactivated
04	Chuck Vacuum	0 or 1	0	0	Sent if chuck vacuum status is changed from command or panel
05	QuietStatus	0 or 1	0	0	Sent if quiet status is changed from command or panel
07	Micro Light ON/ OFF	0 or 1	0	0	Sent if scope illumination output is toggled
08	New lens selected	Lens number	0	0	Sent if lens are changed (Turret Option)
09	CBox reset	S or H	0	0	Sent to reset the joystick controller
10	Bottom Side Mode	0 or 1	0	0	Sent if the chuck side mode is changed
11	Dark Mode	0 or 1	0	0	Sent if the dark mode of the prober is toggled

12	Operational Mode	P – Protected U – Unprotected	0	0	Sent if the operational mode changed
13	Software Fence	C – Chuck P – Platen S – Scope 1 – Positioner 1 2 – Positioner 2 3 – Positioner 3 4 – Positioner 4 5 – Positioner 5 6 – Positioner 6	AUX ID	0	Sent if the software fence of the current stage was updated
14	Active Stage	C – Chuck P – Platen S – Scope 1 – Positioner 1 2 – Positioner 2 3 – Positioner 3 4 – Positioner 4 5 – Positioner 5 6 – Positioner 6	0	0	Sent if the joystick controller stage changed
16	Active Speed	C – Chuck P – Platen S – Scope 1 – Positioner 1 2 – Positioner 2 3 – Positioner 3 4 – Positioner 4 5 – Positioner 5 6 – Positioner 6	1 – Jog speed 2 – Speed 2 3 – Speed 3 4 – Speed 4 X – Index	0	Sent if the current speed of the joystick controller changed
20	System IO	E – EMO Sensor L – LoaderDoor M – MotorPower P – GlobalPressure V – GlobalVacuum A – Cassette 1 B – Cassette 2	0 – Signal low 1 – Signal high	0	The signal level of a system input or output has changed. The new level is notified.
21	Edge Sensor	1 – EdgeSensor 1 2 – EdgeSensor 2	0 – Closed 1 – Open	0	An edge sensor has generated a signal
22	Wafer Vacuum	1 or 0	0	0	The wafer vacuum sensor detected a change. The new level is notified.
23	LoaderGate	L – Load/Open U – Unload/Closed T – Transfer/Unknown	0	0	The state of the loader gate has changed
31	Chuck XY movement	X pos [µm]	Y pos [µm]	AUX ID	Sent during moves and axis shifting (Position Reference Zero)
32	Chuck Z movement	Z pos [µm]	O – Overtravel C – Contact A – Alignment S – Separation T – Search 0 – Undefined	AUX ID	Sent during moves and axis shifting (Position Reference Zero)

33	Chuck move to load pos	L – Load T – Transfer U – Unload	0	0	Sent if chuck is in load position or moves out of load position
34	Chuck new index	New X index [µm]	New Y index [µm]	AUX ID	Sent if new index is set (width of one index step)
35	Chuck new home pos	New X home [µm]	New Y home [µm]	AUX ID	Sent if new home is set (Position Reference Zero)
36	Chuck contact set/reset	Z pos [µm]	S Set R Reset	AUX ID	Sent if a contact height is set or reset
37	Chuck Lift	0 or 1	0	AUX ID	0 is upper, 1 is lifted = down position
39	Chuck new Z height	Height [µm]	A – Align O – Overtravel S – Separation T – Search	AUX ID	A new Z height for the chuck stage was set
41	Scope XY movement	X pos [µm]	Y pos [µm]	0	Sent if move is finished (Position Reference Zero)
42	Scope Z movement	Z pos [µm]	O – Overtravel F – Focus A – Alignment S – Separation 0 – Undefined	0	Sent during moves and axis shifting (Position Reference Zero)
44	Scope new index	New X index [µm]	New Y index [µm]	0	Sent if new index is set (width of one index step)
45	Scope new home pos	New X home [µm]	New Y home [µm]	0	Sent if new home is set (Position Reference Zero)
46	Scope focus set/reset	Height [µm]	S – Set R – Reset	0	Sent if a focus height is set or reset
47	Scope Lift	0 or 1	0	0	0 is lowered, 1 is lifted position
49	Scope new Z height	Height [µm]	A – Align S – Separation	0	A new Z height for the scope stage was set
51	Probe XY movement	X pos [µm]	Y pos [µm]	1,2,3,4,5,6	Sent during moves and axis shifting (Position Reference Zero)
52	Probe Z movement	Z pos [µm]	O – Overtravel C – Contact A – Alignment S – Separation 0 – Undefined	1,2,3,4,5,6	Sent during moves and axis shifting (Position Reference Zero)
54	Probe new index	New X index [µm]	New Y index [µm]	1,2,3,4,5,6	Sent if new index is set (width of one index step)
55	Probe new home pos	New X home [µm]	New Y home [µm]	1,2,3,4,5,6	Sent if new home is set (Position Reference Zero)
56	Probe contact set/reset	Z pos [µm]	S – Set R – Reset	1,2,3,4,5,6	Sent if a contact height is set or reset
57	Probe Lift	0 or 1	0	1,2,3,4,5,6	0 is lowered, 1 is lifted position

59	Probe new Z height	Height [µm]	A – Align O – Overtravel S – Separation	1,2,3,4,5,6	A new Z height for a positioner stage was set
61	Theta Move	Theta pos [deg]	0	0	Position Reference Zero. Sent during moves and axis shifting.
65	Theta new home pos	New Theta home [deg]	AUX ID	0	Sent if new home is set (Position Reference Zero)
90	Configuration Update	0 – Stage independent C – Chuck P – Platen S – Scope 1 – Positioner 1 2 – Positioner 2 3 – Positioner 3 4 – Positioner 4 5 – Positioner 5 6 – Positioner 6	AUX ID	0	An infrequent configuration change for the stage has occurred. Examples: AxisReverse, AxisDisable, JogSize
91	ManualMode	Enabled	Unused	Unused	
100	TemperatureChuck temperature changed	Actual temperature [Celsius]	Target temperature [Celsius]	Dew point temperature [Celsius]	Sent after temperature changes occur
101	TemperatureChuck status changed	Status: U – Unknown H – Heating C – Cooling A – AtTemperature R – HoldReady O – Holding S – Standby T – Soaking	Soak timer value [minutes]	Error ID	Sent after status changes, soak time updates, or internal temperature chuck errors, which are not bound to a command
102	TemperatureChuck options changed	0	0	0	Sent after temperature chuck option updates
1130	New active AUX site	AUX ID	0	0	Sent after transfer move between AUX sites

ASCII Strings in Notifications

ASCII string in numeric representation as used in notifications:

Character	ASCII code
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

Chapter

10 Error Codes

The following error sets are included in this section:

- Kernel Errors
- Server and Common Errors
- Application Errors

Kernel Errors

Code	Description	Notes
0	No Error	
2	Checksum error	The serial communication is distributed or the hardware is defective.
3	Unknown command number	
5	Bad data format	
6	Transmission timeout	
8	Command not implemented	
9	Command timeout	
10	System option disabled or not present	The feature does not exist or is disabled.
11	Not supported	
12	Too many commands	32 commands can run parallel.
13	IO16 disabled or not present	
14	Stage disabled or not present	The stage does not exist or is disabled.
15	Axis disabled or not present	The axis does not exist or is disabled.
16	Edge sensor disabled or open	The edge sensor is enabled and not connected or it is triggered.
17	Not supported by the lift	A pneumatic lift can't initialize or move to a variable position.
18	Conflict between brake and lift	The lift and the brake are enabled at the same time.
19	Command task exception	
20	Initialization failed	The initialization had a general technical failure. It can be a none-triggered end limit switch.
21	Unexpected software failure	The firmware has an abnormal program termination. That needs a reinitialization.
22	Unexpected hardware failure	The hardware has lost the power supply. That needs a reinitialization.
23	Couldn't leave the end limit	A special error of the initialization. The stage couldn't leave the end limit in the second step. The initialization distance may be too small.
24	IO16 function disabled	

25	Not initialized	
26	Stage locked by hardware	
27	Stage locked by software	
28	Invalid position	
29	No focus height set	The microscope set focus height is required for ScopeFollowMode
30	Alignment is not present	
31	Matrix compensation is not present	
32	Z profile is not present	
33	Compensation is not present	
34	Thermo compensation is not present	
39	Position already occupied	
40	Contact search height violation	The destination is beyond the searching height.
41	Invalid scaling factors	
42	Invalid chuck system	The current chuck frame system of BEP300 doesn't correspond to the machine type
43	This value is too small	
44	This value is too large	
45	Data iterator at end position	
46	Invalid data iterator	See command *GetDataIterator.*
47	Could not save configuration	
48	Data not found	The parameter path or name is invalid.
49	Permission denied	This parameter is read only.
50	Invalid data	
51	Cover is open	
52	No vacuum	No system vacuum was detected.
53	No compressed air	No system compressed air was detected.
54	Vacuum Timeout	
55	Contact height violation	The target position is above the current contact height.
56	EMO is active	If the system has EMO button detection hardware, the reason for the motor power supply's absence can be classified. If the system has no EMO button detection hardware, misconfiguration could result.
57	No motor power	The motor power supply is open. Open cover or EMO circuit.
58	Compressed air timeout	The compressed air on the air bearing couldn't be detected after the time limit.
59	Load or Unload cycle not completed	ProbeShield and pin chuck machines are allowed to move to the load position except the commands *MoveChuckTransfer*, *MoveChuckLoad* and *InitChuck.*
61	Stage at contact height	
62	Contact height not set	
63	Invalid Z profile data	
64	Parfocality adjustment failed	

65	Not allowed in Back side Mode	The command *MoveChuckLoad* isn't allowed in the bottom side mode.
66	Stage is busy	
67	Alignment angle too large	The alignment angle limit is +/- 45°.
68	Invalid jog value	The jog size is zero or smaller.
69	Invalid index value	The index size is zero or smaller.
70	Invalid table name	
71	Table not found	
72	Invalid table point	
73	Table point already exists	
74	Table already exists	
75	Software fence error	The destination of the move is outside of the move area.
76	Table point not found	
77	Scope camera interlock engaged	
78	Software fence cannot be crossed	The path of the move crosses the fence. That means a part of the path is outside of the move area.
79	Critical position error	
80	Tracking window error	The current tracking error was too large and the movement aborted.
81	Invalid end limit state	Both end limit switches are triggered.
82	Axis at positive end limit	
83	Axis at negative end limit	
84	Axis on software fence	The current position is on the software fence. The stage will try to move outside of the move area.
85	Chuck camera height violation	The destination is beyond the camera height and the chuck camera is docked.
86	Invalid software fence	The current software fence is invalid after clipping of the software limits. The size is zero or the fence is wrapped.
87	Camera docking error	The sensors couldn't detect the camera on any position. The camera position is unknown.
88	Movement stopped by edge sensor	The edge sensor was triggered and has stopped all motions of the stage.
89	Stage movement stopped	A stop command has stopped the current movement.
90	Contact not found	The edge sensor hasn't triggered.
91	Movement denied by positioner interlock	
92	No contact height is set for positioner	For the positioner follow mode a contact height for the positioner must be set
93	Chuck camera interlock engaged	The sensor has detected an approach of the chuck to the camera in Z direction.
94	At contact search height	The current position is searching height. The stage will try to cross that height.
95	At chuck camera height	The current position is camera height. The stage will try to cross that height.
96	Simultaneous init not allowed	
97	Loading carriage interlock engaged	

98	Loading carriage is missing	
99	Loading carriage is in the machine	
110	IO16 Input error	
111	IO16 Output error	
112	IO16 Could not set irq mask	
113	IO16 Output cannot switch	
114	IO16 Invalid channel	
115	No data trace	The PMD hasn’t recorded data
116	Move wait abort	
119	PMD task exception	
120	PMD parameter transfer error	
121	PMD update error	
122	PMD interrupt error	
123	PMD zero motion	
124	PMD invalid tracking window	The value for the tracking window is too large or too small
125	PMD motion values error	The value for the motion error is too large or too small
126	PMD invalid parameter bank	
127	Edge sensor error	No access to the edge sensor device
130	Device disabled or not connected	
131	Cannot access the device	
132	Dew point sensor disabled or not connected	
133	Heating or cooling process aborted	
134	Hold mode requires HoldReady state	
135	Temperature value out of bounds	
136	Device currently in power save mode	
137	Internal temperature chuck device error	
138	Chuck is not in room temperature range	
151	Nucleus Kernel not initialized	
152	Nucleus Kernel already initialized	
153	Nucleus Kernel Component error	
154	Nucleus Kernel Component stop	
155	Nucleus Kernel unsupported stage error	
156	Nucleus Kernel stage open error	
157	Nucleus Kernel Move error	
158	Nucleus Kernel stage configure error	
159	Nucleus Kernel Override Function error	
160	Nucleus Kernel Component locked	
161	Nucleus Kernel set velocity error	
162	Nucleus Kernel read position error	

B&R Controller Errors

Code	Description
201	Unexpected TCP receive error
202	Unexpected TCP send error
203	Unknown SNI command
210	Null pointer in command list
211	Unexpected axis error
212	Control switch on timeout error
213	Moving timeout error
214	Axis state error
215	Power off timeout error
216	Power on timeout error
217	Stop timeout error
218	Timeout acknowledge error
219	Timeout move ready axis error
220	Timeout busy
221	Move axis error
222	Init axis error
223	OS version error
224	Stop axis error
225	Quiet axis error
226	Function call error
227	Homing timeout axis error
228	Homing abort error
229	Axis read state error
230	Fatal Configuration Error
231	Axis Error
232	Axis SetDatum Error
233	Axis GetDatum Error
234	Axis Set/Get Datum Parameter Null Error
235	Axis SetDatum Timeout
236	Axis GetDatum Timeout
237	Axis StandStill Timeout
238	Axis Set/Get Datum Parameter Compare Error
239	No Axis Power or Axis Config is loading - wait if Motor Power is on
240	Loading Axis Config Item from File failed
241	No Axis Motor Power - e.g. EMO pressed, Cover open...
242	Controller reboot is required to take over the changed parameters.
243	Axis Scan Error
244	Analog IO Error
245	Command not supported for B&R Wiring Tester Config

Server and Common Errors

	Code	Description
Server Errors	500	Command Acknowledge Timeout
	501	Command Response Timeout
	502	Invalid Command Format
	503	Invalid Response Format
	504	Checksum Failure
	505	Out of Node IDs
	506	Emergency Cancel
	507	Too Many Active Commands
	508	Serial Line Error
	509	Command Number Not Found in Command File
	510	Command ID lost in DoProberCommand
	511	Process Not Registered With RegisterProberApp
	512	Destination Not Found for Command
	513	Open Port Failed
	514	Not Connected to Host
	515	Invalid String - unable to parse (unknown Structure)
	516	Application already registered
	517	No Source for Response
	518	Command ID not in List
	519	Invalid Buffer Size (NULL)
	520	Send to Socket has failed
	521	Command Unknown - not in Demo Response Table
	522	KernelClient or DemoClient not registered
	523	Out of Command IDs
	524	Empty Command String in DoProberCommand
	525	Waiting for Response broken
	526	Kernel Commands not available
	527	Parameter convert Error
	528	Parameter not implemented
	529	Stage or Axis not available
	530	Hardware not available
	531	No Controller for Command
	532	Velox License is not valid
	533	Invalid Command Parameter(s)
	534	Movements not possible (B&R upgrade required)
	535	No Socket Connection
	536	Ping Failed
Common Command Handler Errors		
	550	Dialog Already Opened
	551	Prober Not Paused
	552	Incorrect File Name
	553	Functionality not implemented
	554	Save project as operator not allowed
	560	Unknown command

Application Errors

	Code	Description
Misc. System Errors		
	600	Unknown Command
	601	Command Not Implemented
	602	Not in Remote Mode
	603	Resumed when not Paused
	604	Already in Remote Mode
	605	No Current Project
	606	Project Not Found
	607	Unable to Initialize Position Table
	608	Unable to Move to Site
	609	Stage not Present
	610	Unable to Enter Bin Code
	611	Invalid Site Bin Assignment
	612	No Yield Data
	613	Unable to Read Yield Data
	614	End of Site Table
	615	Timeout During Command Execution
	616	Unable to exit RFCal mode
Wafer Map Errors		
	700	No dies are selected
	701	Target die was not synchronized
	702	Not in external mode
	703	End of wafer
	704	Die indices are not valid
	705	“StepFirstDie” has not been issued
	706	Not a valid bin number
	707	No die currently being probed
	708	Inking of die failed
	709	Communication problem with DDE Server
	710	Not a valid bin assignment
	711	“StartInkerRun” has not been issued
	712	Wafer Map processing another move request
	713	WaferMap paused or not running
	714	Sub Die index is not valid
	715	Illegal Parameters
	716	Cluster indices are not valid
	717	Clusters are disabled
	718	No current Position
	719	Command not Implemented
	720	No Wafer Points
	721	The processing was cancelled
	722	Failed to profile due to many failed dies

Optical Control Errors		
	750	Communication error device
	751	Illegal Parameters
	752	A-Zoom Busy
	753	Motorzoom Error
	754	Command not supported
	755	No support by the equipment
	756	Communication error Velox
Motion Viewer Errors		
	760	Trace already Running
	761	No trace data
	762	Saving trace results failed. Sub description is appended, ex. (invalid absolute file name.)
Z-Profiling Errors		
	800	Z-Profiling is busy
	801	Z-Profiling is processing another command
	802	Command parameter is missed
	803	File could not be saved
	804	File could not be opened
	805	Illegal parameters
	806	'StepFirstZProfilePoint' has not been issued
	807	End of the profile
	808	The processing was cancelled
	809	The profile is empty
	810	Not an automatic mode
Spectrum Errors		
	850	Model Not Trained
	851	Position not trained
	852	Model Not Found
	853	Focus Not Found
	854	Invalid Parameters
	855	Spectrum Busy
	856	Possible Confusion
	857	Could Not Train Model
	858	Inspect mode not trained
	859	Could not access data
	860	Wizard is already running
	861	Edge Not Found
	862	Maximum Number Exceeded
	863	Acquisition Error
	867	No valid file name
	868	No valid camera
	869	No valid bitmap
	870	Could not save bitmap
	871	Average Image not generated
	872	Piezo out of range

	873	Wrong parameter number
	874	Internal Error Spectrum Vision System
	875	Tool without dedicated mount position
	880	String Not Recognized
	881	Bar Code Not Recognized
	882	DataMatrix Code Not Recognized
	890	ProbeCard OCS Failure
	891	ProbeCard Alignment Failure
	892	PreAlignment Failure
	893	Lens is not calibrated
	894	Alignment Failure
	895	Automation not active
	896	Automation failure
TTL Interface Errors		
	910	Time to test is elapsed. A timeout occurred while testing. The test was aborted due to this timeout. This error occurs in case the test system is not responding as expected. If the test takes too much time, increase the timeout to fix it.
	911	ME Board input error. Raised in case of errors related to reading bits from the DIO card.
	912	ME Board output error. Raised in case of errors related to writing bits from the DIO card.
	913	Test canceled due to user interaction or an error.
	914	TTL test in progress. This error is caused if sending a Velox Command to the TTL interface while another one is being processed.
	915	Illegal Parameters. The Parameters submitted to SetTTLLine reference a non existing line or bit.
	916	This is an internal error that occurred in the TTL interface. The user can't do anything to fix it.
	917	Data overflow. The Tester is sending more EOT-Signals than expected according to the current cluster settings. Raised only if checking of the Busy-Line is enabled in the Advanced Settings View.
	918	Data underflow. The Tester is sending fewer EOT-Signals than expected according to the current cluster settings. Raised only if checking of the Busy-Line is enabled in the Advanced Settings View.
Database		
	920	Database error
	922	Probecard does not match
	923	Probecard touchdown exceeded
	924	Wafer ID does not match
	925	Lot not allowed
Tester		
	930	Invalid parameters
	931	No valid project
	932	Project database corrupted
	933	No valid Lot/Tile ID
	934	Internal tester error
ReAlign Errors		

	1200	ReAlign failed
	1201	ReAlignWizard busy
VeloxPro Errors		
	1400	Recipe error
	1401	Invalid data
	1402	Substrate error
	1403	Handling error
	1404	Illegal parameter
	1405	Host communication error
	1406	Internal error
	1407	Tester error
	1408	Cassette not placed
	1409	Cassette not docked
	1410	Prober action not allowed
Scripting Console Errors		
	3000	Alias is not found
	3001	Program running another script
	3002	Illegal Parameters
Kernel Setup Errors		
	Command Reads/Writes many Datums. In these cases, the Command returns the first occurred Error. Permission denied Errors (Kernel Error 49) are not considered.	
	3030	KernelSetup is busy
	3031	Datum not Found
	3032	Internal Datum convert Error
	3033	Unknown Datum Type (TypeCode)
	3034	Could not work in DemoMode
	3035	Invalid Parameter
	3036	Kernel Data not loaded
	3037	File Data not loaded
	3038	Could not create Folder
	3039	File does not exist
	3040	Unknown Command

Appendix

A Legacy Commands: SCPI

The Standard Commands for Programmable Instruments (SCPI) are described in this section.

- Summary of Commands
- SCPI Device IDs
- SCPI Descriptions

NOTE

SCPI can be handled by Velox when using the GPIB or DDE interface. SCPI is not supported in Scripting Console or through TCP/IP.

Summary of Commands

The SCPI command set is legacy and should only be used for backwards compatibility. Test executives should be developed using SCI whenever possible.

Command	Description
`:align:wafer:busy?`	Returns the current state of the wafer alignment.
`:align:wafer:cancel`	Cancels the alignment process before it completes.
`:align:wafer:start`	Starts an auto wafer alignment process.
`:automation:run:temperature:cancel`	Stops the currently running temperature transition operation.
`:automation:run:temperature:start`	Starts an automated temperature transition to the desired temperature.
`:automation:run:temperature:status?`	Returns the status of the currently running temperature transition operation.
`:automation:set:active`	Enables or disables the usage of automation features such as Auto XY, Auto Z, and Probe tip tracking.
`:automation:set:active?`	Returns "ON" or "OFF" to indicate whether automation is currently active.
`:capture:video`	A fully qualified file name that defines where the captured video will be saved.
`:move:absolute`	Moves the device z-axis to a safe position; then moves the x- and y-axis to specified coordinates, and then moves z-axis to a specified position.
`:move:absolute?`	Returns the device's current position in user coordinates.
`:move:absolute:chuck`	This command is the same as the `:mov:abs` command.
`:move:absolute:options`	Allows the user to specify the type of coordinates being used in specifying the x-, y-, and z-axis positions.
`:move:align`	Moves the device to the alignment position.
`:move:center`	Moves the device to the center (0,0) of the device-movement area.
`:move:contact`	Moves the device to contact the wafer.
`:move:contact?`	The query returns TRUE if the device is at contact position and FALSE if not.

Command	Description
`:move:down`	Lowers the device to a predefined down position. Use `:mov:sep` instead of this command.
`:move:down?`	The query returns TRUE if the device is down and FALSE if the device is up.
`:move:kill`	Used to stop motion on the specified device.
`:move:load`	Moves the device to the load-device position. The load position is either the front-center or front-left corner, depending on your model of probe station.
`:move:microscope:absolute`	Moves the fine focus stage of the eVue microscope to an absolute position.
`:move:microscope:absolute?`	Returns the current stage location of the eVue fine focus stage.
`:move:microscope:focus`	Performs an Auto-Focus operation.
`:move:probeplan:absolute:die`	Moves the device to the x, y die index position, as defined in the active probe plan file.
`:move:probeplan:absolute:die?`	The query returns the current probe plan x, y indexes.
`:move:probeplan:absolute:index`	Moves the device to the nth die site. Only sites designated for testing in the active probe plan file are counted.
`:move:probeplan:absolute:index?`	Returns the die-site number corresponding to the current device position.
`:move:probeplan:absolute:index:label`	Moves the device to a die/subsite location.
`:move:probeplan:absolute:index:subsite`	Moves the device to a specified die and subsite, using the testable die index and the numeric index for the subsite.
`:move:probeplan:absolute:location`	Moves the device to an x, y die index and subsite location in one move.
`:move:probeplan:absolute:subsite`	Moves the device to the specified subsite index position, as defined in the active probe plan file.
`:move:probeplan:absolute:subsite?`	Returns the current probe plan subsite indexes.
`:move:probeplan:absolute:subsite:label`	Moves the device to the subsite with the given label on the current die.
`:move:probeplan:first:die`	Moves the device to the first die site designated for testing in the active probe plan file.
`:move:probeplan:first:subsite`	Moves the device to the first subsite on the current die designated for testing in the active probe plan file.
`:move:probeplan:last:die`	Moves the device to the last die site designated for testing in the active probe plan file.
`:move:probeplan:last:subsite`	Moves the device to the last subsite designated for testing in the active probe plan file.
`:move:probeplan:next:die`	Moves the device to the next die site designated for testing in the active probe plan file.
`:move:probeplan:next:site`	Moves the device to the next site designated for testing in the active probe plan file. The site can be either a die site or a subsite, whichever comes next.
`:move:probeplan:next:subsite`	Moves the device to the next subsite on the current die designated for testing in the active probe plan file.
`:move:probeplan:prior:die`	Moves the device to the prior die site designated for testing in the active probe plan file.
`:move:probeplan:prior:site`	Moves the device to the prior site designated for testing in the active probe plan file. The site will be either a die site or a subsite, whichever came last.
`:move:probeplan:prior:subsite`	Moves the device to the prior subsite on the current die designated for testing in the active probe plan file.

Command	Description
:move:probeplan:relative:die	Specifies relative moves within the active probe plan wafer map.
:move:probeplan:relative:index	Specifies relative moves within the active probe plan wafer map.
:move:relative	Moves the device z-axis to a safe position; moves to the specified x- and y-axis positions, then moves the z-axis to a specified position. Moves are made relative to the current position.
:move:relative:chuck	This command is the same as the :move:rel command.
:move:separate	Moves the device to break the contact between the probes and DUT.
:move:separate?	The query returns TRUE if the device is at contact position and FALSE if the device is retracted.
:move:up	Raises the device.
:move:up?	The query returns TRUE if the device is up and FALSE if it is down.
:probeplan:absolute:reference	Moves the reference die to the specific X, Y location without moving the origin at the same time. It is different from :prob:ref which does move the origin when a new location is specified.
:probeplan:align	Sets the alignment position for a probe plan.
:probeplan:die:location?	Returns the column row in wafer map user coordinate space of the given testable index number.
:probeplan:die:size	Sets the die size of the currently loaded wafer map file.
:probeplan:die:xsize?	Returns the horizontal size of the die in the currently loaded wafer map.
:probeplan:die:ysize?	Returns the vertical size of the die in the currently loaded wafer map.
:probeplan:flatnotch	Sets the FLAT or NOTCH on a wafer map.
:probeplan:flatnotch?	Returns a string that contains two values: Type and Size. Type is either FLAT or NOTCH and the Size depends on the Type.
:probeplan:grid:shift	Shifts the die pattern on the wafer map by X, Y microns.
:probeplan:grid:shift?	Returns the current shift of the die pattern on the wafer map.
:probeplan:load	Loads and activates a probe plan file.
:probeplan:load?	The query returns the current probe plan filename, if any.
:probeplan:mark:die	Specifies the color used to identify die sites when you display a wafer map.
:probeplan:mark:xydie	Similar to the :prob:mark:die command, but uses x, y coordinates to specify die location.
:probeplan:nsubsites?	Returns the number of subsites marked for testing in the active probe plan file.
:probeplan:ntested?	Returns the number of die marked for testing in the active probe plan file.
:probeplan:origin	Used to set the origin for the currently loaded wafer map.
:probeplan:origin?	Queries the currently loaded wafer map for the origin die.
:probeplan:quality:size	Sets the quality area of a wafer.
:probeplan:quality:size?	Returns the quality area size of the current wafer map.
:probeplan:reference	Sets the reference die for the currently loaded wafer map.
:probeplan:reference?	Queries the reference die for the currently loaded wafer map.
:probeplan:save	Saves the currently loaded wafer map file from memory onto disk.
:probeplan:set:palette	Defines the colors in the palette.
:probeplan:set:subsite	Defines a subsite location (device position relative to the die site).

Command	Description
:probeplan:set:subsite:label:active?	Returns the test status of a subsite at the given index.
:probeplan:show	
:probeplan:street:size	This command sets the street size of the current wafer map in X and Y.
:probeplan:street:xsize?	Returns the X street size of the current wafer map.
:probeplan:street:ysize?	Returns the Y street size of the current wafer map.
:probeplan:subsites:delete	Deletes the subsite at the specified index.
:probeplan:subsites:delete:all	Deletes all of the subsites in the subsite list.
:probeplan:subsites:label?	Returns the descriptive text field used to identify the subsite.
:probeplan:subsites:total?	Returns the total number of subsites whether they are marked as active or not.
:probeplan:subsites:xoffset?	Returns the X offset associated with a given subsite.
:probeplan:subsites:yoffset?	Returns the Y offset associated with a given subsite.
:probeplan:test:die	Marks die to test for the currently loaded wafer map.
:probeplan:test:die?	Queries whether die was marked to test for the currently loaded wafer map.
:probeplan:testing:sequence	Defines the stepping sequence of the currently load wafer map.
:probeplan:testing:sequence?	Returns a string describing the current stepping pattern.
:probeplan:wafer:orientation	Sets the diameter of the currently loaded wafer map.
:probeplan:wafer:orientation?	Returns the diameter of the currently loaded wafer map.
:probeplan:wafersize	Sets the orientation of the current wafer map.
:probeplan:wafersize?	Returns the orientation of the current wafer map.
:probeplan:xy:orientation	Sets the orientation of the X and Y axes in the wafer map.
:probeplan:xy:orientation?	Queries the positive direction of the x- and y-axis of the currently loaded wafer map.
:reverse:orientation	This command reverses the polarity of the user coordinate system. The reversal only applies to the x- and y-axes.
:set:auxiliary	Turns power to the probe station auxiliary power outlet on or off.
:set:busy?	Checks to see if the probe station is currently performing any operations.
:set:chamber:purge	Controls the amount of MicroChamber air purge (OFF, MANUAL, QUICK, AUTO).
:set:chamber:purge?	Returns the current MicroChamber air purge setting.
:set:compensation	Turns compensation mode on or off.
:set:compensation:factor	Numerically sets the compensation factors for movement along the x- and y-axes.
:set:compensation:mode	Numerically sets the compensation factors for movement along the x- and y-axes
:set:contact	Sets the contact location for the Z stage and positioners. It defines the location that the DUT comes in contact with the probes
:set:contact?	Returns the contact position of the z stage.
:set:contact:active	Turns the contact mode on and off. Optional parameters can be used to enable or disable dialog prompts. A value of “ON” enables prompts to be displayed. This is the default if no parameter is provided. “OFF” disables all prompting for setup of Z-Profile.
:set:contact:active?	Returns ON or OFF depending on contact status.
:set:contact:band	Sets the contact search band for contact mode.
:set:contact:band?	Returns the currently set contact/search band used in contact mode.

Command	Description
:set:contact:mode	Turns on different contact modes.
:set:contact:mode?	Returns values for the different contact modes.
:set:contact:speed	Sets the search speed used for the Z-axis in contact mode.
:set:contact:speed?	Returns the current search speed for Z-axis that is used in contact mode.
:set:contact:use:zprofile	Enables Z-Profile when programmable contact mode is activated.
:set:contact:use:zprofile?	Returns ON if Z-Profile is enabled in programmable contact mode, otherwise returns OFF.
:set:delay	Specifies a time in milliseconds for the probe station to delay after each operation.
:set:delay?	Returns the current delay time.
:set:edgesense	Turns edge sense mode on or off.
:set:edgesense?	Returns the status of contact mode: enabled or not enabled.
:set:edgesense:overdrive	Sets the distance the chuck rises after contact is made with the edge sense probe.
:set:edgesense:overdrive?	Returns the current overdrive amount that is set in contact mode.
:set:edgesense:switch	Identifies the non-contacted position of the switch in the edge sense probe.
:set:edgesense:switch?	Returns the current switch type that is defined for use with edge sense.
:set:inker	Used to fire the inker.
:set:joystick:enabled	Enables or disables the joystick.
:set:joystick:enabled?	Queries whether or not the joystick is enabled.
:set:joystick:mode	Changes the motion control mode used by the joystick (SCAN, SCANAUTOZ, SCANZ, INDEX, DIE, SUBINDEX, JOG, THETA).
:set:joystick:mode?	Returns the currently-defined motion mode for joystick.
:set:light	Turns the microscope light on or off.
:set:light?	
:set:manual	Toggles from semiautomatic to manual mode.
:set:manual?	Returns ON if the probe station is operating in manual mode and OFF if operating in semiautomatic mode.
:set:microscope:agc	Enables/disables the use of Automatic Gain Control for the microscope.
:set:micr:agc?	Queries the state of Automatic Gain Control for the microscope.
:set:microscope:auto:illuminate	Runs an auto illumination operation at the current location.
:set:microscope:exposure	eVue only. Exposure level in ms from 0.5 to 100.
:set:microscope:exposure?	eVue only. Returns the exposure level.
:set:microscope:zoom	Sets the zoom value used by the A-Zoom microscope. This command also works in the QUAD screen mode.
:set:microscope:zoom?	Returns the zoom value used by the A-Zoom microscope. This command has an optional parameter that can be used to query the different zoom levels of each optical path
:set:preset	Presets the coordinate system coordinates for :move:absolute device movements by assigning the specified user coordinates to the current device location.
:set:salign	Turns on and off software alignment mode.
:set:salign?	Returns ON if the probe station is operating in software alignment mode and OFF if it is not.

Command	Description
`:set:separate`	Sets the separate distance away from contact used in auto z-stepping.
`:set:separate?`	Returns the separate distance of the z-stage.
`:set:station?`	Returns the probe station's device identification number.
`:set:unit`	Specifies the measurement units to be used for all input/output fields.
`:set:unit?`	Returns the unit of measurement that is being used by distances and coordinates.
`:set:usedeviceid`	Controls the use of the device ID parameter for many SCPI commands.
`:set:usedeviceid?`	Queries the current setting of the device ID in use.
`:set:vacuum`	Turns the central chuck vacuum on or off. Does not affect vacuum on corners or auxiliary devices.
`:set:vacuum?`	Returns ON or OFF depending on the state of the vacuum output line (not a sense line).
`:set:vacuum:auxiliary`	Turns vacuum on or off to the auxiliary chucks. Elite 300 only.
`:set:vacuum:auxiliary?`	Returns ON or OFF depending on the vacuum control line for the requested auxiliary chuck. Elite 300 only.
`:set:vacuum:ring`	Enables vacuum to the rings of the chuck. Elite 300 only.
`:set:vacuum:ring?`	Returns ON or OFF depending on the enable state of the specified vacuum ring. Elite 300 only.
`:set:velocity`	Specifies the velocity for device movement along each axis. Does not apply to moves made in scan mode.
`:system:beep`	Causes the system to emit a sound.
`:system:configuration?`	Returns a string response containing the system configuration value of the parameter requested.
`:system:delay`	Makes the probe station wait the specified length of time before executing the next instruction.
`:system:display`	Displays a one-line message on the probe station screen.
`:system:error?`	Returns the last error recorded by the probe station.
`:system:identification?`	Returns the probe station identification string.
`:system:limits:travel?`	Returns the limits of travel for the specified channel.
`:system:operation:mode`	Sets Velox software to REMOTE (all windows but video are hidden) or LOCAL (normal UI operations) mode.
`:system:operation:mode?`	Returns REMOTE or LOCAL depending on what mode is currently active.
`:system:platen?`	Returns the status of the platen arm.
`:system:vacuum:sense?`	Returns ON if the system senses vacuum, and returns OFF if it does not sense vacuum.
`:system:version?`	Returns the probe station software release version.
`:thermalchuck:activate`	Sets the thermal chuck to be active.
`:thermalchuck:activate?`	Checks to see that the thermal chuck is active.
`:thermalchuck:deactivate`	Deactivates the thermal and sets it to idle.
`:thermalchuck:status?`	Returns the status of the thermal chuck.
`:thermalchuck:temperature:current?`	Returns the current temperature of the thermal chuck in degrees Celsius.
`:thermalchuck:temperature:settemp`	Sets the thermal chuck to a temperature.
`:thermalchuck:temperature:target?`	Returns the current target temperature.
`:thermalchuck:temperature:window`	Sets the window range of the thermal chuck.

Command	Description
`:thermalchuck:temperature:window?`	Returns the current window range.
`:vision:search:target`	Performs a pattern req. search using Spectrum Vision.
`:vision:set:match:score`	Sets the threshold score used during vision search operations.
`:vision:train:target`	Trains a target using the region defined by the vision box in the video window.

SCPI Device IDs

DevID	Description
2	Used to control the chuck on Elite Series, Summit 12000-Series, S300 and Alessi 6100
3	Microscope channel
7	Left #1 Motorized positioner
8	Right #1 Motorized positioner
9	Left #2 Motorized positioner
10	Right #2 Motorized positioner
11	Left #3 Motorized positioner
12	Right #3 Motorized positioner

SCPI Descriptions

The command descriptions provide details regarding individual commands. Short commands, descriptions, parameters, examples and related commands are described as appropriate to each command. Parameters must be separated by exactly one (1) character space.

Alignment Commands

Automation Commands

Capture Video Command

Move Commands

Wafer Map Commands

Reverse Orientation Command

Set Commands

System Commands

Thermal Chuck Commands

Vision Commands

Alignment Commands

Commands		
:align:wafer:busy?	:align:wafer:cancel	:align:wafer:start

:align:wafer:busy?

Short Command

`:alig:waf:busy?`

Description

Returns the current state of the wafer alignment. An alignment can be started with :align:wafer:start. Possible return values are: SUCCESS, ERROR, BUSY. SUCCESS is returned when the wafer alignment procedure has been completed successfully. ERROR is returned if there was an error during the alignment. BUSY is returned if the alignment process is still working.

Parameters

None

Example

```
:align:wafer:start
COMPLETE
:align:wafer:busy?
SUCCESS
```

Related Commands

:align:wafer:cancel
:align:wafer:start

:align:wafer:cancel

Short Command

`:alig:waf:canc`

Description

Once an alignment is started with the command :align:wafer:start, it can take several minutes to complete. The command `:align:wafer:cancel` can be used to stop the alignment process before it completes.

Parameters

None

Example

```
:align:wafer:start
COMPLETE
:align:wafer:cancel
COMPLETE
```

Related Commands

:align:wafer:busy?
:align:wafer:start

:align:wafer:start

Short Command

`:alig:waf:star`
Use Die Size

Description

Starts an auto wafer alignment process. When this command is sent, the vision train box in the video window must be on the repeating target image for the die. An auto alignment process can take a long time, so this command returns as soon as alignment starts. The status can be checked with `:align:wafer:busy?` and it can be canceled with `:align:wafer:cancel`.

Parameters

Use Die Size (optional)

True – If true, then the die size from the wafer map is used for alignment.

False – If false, then it uses small steps to find the repeating pattern.

Example

```
:align:wafer:start
COMPLETE
```

Related Commands

:align:wafer:busy?
:align:wafer:cancel

Automation Commands

Commands	
:automation:run:temperature:cancel	:automation:set:active
:automation:run:temperature:start	:automation:set:active?
:automation:run:temperature:status?	

:automation:run:temperature:cancel

Short Command

`:aut:run:temp:canc`

Description

This command is used to stop the currently running temperature transition operation. It returns "COMPLETE" if the operation was stopped successfully. If the operation encounters an error, an error string will be returned.

Parameters

None

Example

```
:automation:run:temperature:cancel
COMPLETE
```

Related Commands

:automation:run:temperature:start
:automation:run:temperature:status?

:automation:run:temperature:start

Short Command

`:aut:run:temp:star`

Description

Given the currently loaded automation data setup and a temperature value, this command starts an automated temperature transition to the desired temperature. It returns immediately because this can be a 1+ hour operation. The status of the transition can be monitored with `:automation:run:temperature:status?`

Parameters

Temperature

Floating point number that defines the new desired temperature.

Example

```
:automation:run:temperature:start 200.0
COMPLETE
```

Related Commands

`:automation:run:temperature:status?`

:automation:run:temperature:status?

Short Command

`:aut:run:temp:stat?`

Description

This command returns "COMPLETE" if the temperature transition operation has completed successfully. It returns “BUSY” if the operation is still working. If the operation encounters an error, an error string with ‘@’ as the first character will be returned. The error string contains specific information on the type of error.

Parameters

None

Example

```
:automation:run:temperature:status?
BUSY
```

Related Commands

`:automation:run:temperature:start`

:automation:set:active

Short Command

`:aut:set:act`

Description

This command enables or disables the use of automation features such as Auto XY, Auto Z, and Probe tip tracking when moving from die to die in a wafer map.

Parameters

ON – Enables the use of automation features

OFF – Disables the use of automation features

Example

```
:automation:set:active ON
```

Related Commands

`:automation:set:active?`

:automation:set:active?

Short Command
`:aut:set:act?`
Description
This command returns ON or OFF to indicate whether automation is currently active.
Parameters
None
Example
`:automation:set:active?` `ON`
Related Commands
`:automation:set:active`

Capture Video Command

Commands
:capture:video

:capture:video

Short Command
`:capt:vid` *Filename*
Description
Captures the current live video frame and saves it to disk as a .bmp file. The file name must be a fully qualified path and file name. To include spaces in the file name, enclose it in quotes such as C:\Program Files\Cascade\Velox\UserData\Video Image.bmp.
Parameters
Filename A fully qualified file name that defines where the captured video will be saved.
Example
The following example captures the current video frame and saves it to the specified location. `:capt:vid “C:\Program Files\Cascade\Velox\UserData\VideoImage.bmp”.`

Move Commands

Commands		
:move:absolute	:move:microscope:absolute?	:move:probeplan:first:subsite
:move:absolute	:move:microscope:focus	:move:probeplan:last:die
:move:absolute:chuck	:move:probeplan:absolute:die	:move:probeplan:next:site
:move:absolute:options	:move:probeplan:absolute:die?	:move:probeplan:next:subsite
:move:align	:move:probeplan:absolute:index	:move:probeplan:prior:site
:move:center	:move:probeplan:absolute:index?	:move:probeplan:prior:subsite
:move:contact	:move:probeplan:absolute:index:label	:move:probeplan:relative:die
:move:contact?	:move:probeplan:absolute:index:subsite	:move:probeplan:relative:index
:move:down	:move:probeplan:absolute:location	:move:relative:chuck
:move:down?	:move:probeplan:absolute:subsite	:move:separate
:move:kill	:move:probeplan:absolute:subsite?	:move:separate?
:move:load	:move:probeplan:absolute:subsite:label	:move:up
:move:microscope:absolute	:move:probeplan:first:die	:move:up?

:move:absolute

Short Command

`:mov:abs`

The `:move:absolute:chuck` and the `:move:absolute` commands are identical in function.

devID

x

[none]

y

[none]

z

[up]

[down]

[none]

theta

[none]

coordinate space

USER

STAGE

Description

This command moves the device to the specified coordinates. The probe station moves the device's z-axis to a safe position before moving the device in x and y. Then it moves the z-axis to the supplied position after the device reaches the new location. The destination is not affected by wafer alignment, the load angle, the wafer map angle, or the wafer map axis settings.

The coordinate space USER is relative to the location currently defined as the user origin. You can use the `:set:preset` command to change the origin of the coordinate system.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

x

Specifies the x-axis coordinate to which the device will move when this command is executed. This number can be an integer, floating-point, or the character string none. If none, there is no change in the x-axis position.

The actual limits for the value of this parameter depend on the origin of the user coordinate system. The x coordinate value is interpreted as mils or microns. Use the `:set:unit` command to switch between mils and microns.

y

Specifies the y-axis coordinate to which the device will move when this command is executed. The legal values for this parameter are the same as previously described for the x-axis coordinate.

z

Specifies the device height at the end of the move. This number can be an integer, floating-point, or the character strings: up, down, or none. Entering none for the z position will cause the z-axis to return to its present position after separation is ensured for safe x, y, or theta movement.

For probe stations with pneumatic chucks, use one of the following character strings: up, down, or none. If none, the chuck height remains unchanged.

For probe stations with motorized chucks, this parameter can be an integer, floating-point, or the character strings up, down, and none.

theta (optional)

The theta parameter defines the final destination for the theta axis. It is only valid for the chuck channel of motion. Theta is in milli-degrees. This parameter is required to specify the [*coordinate space*] parameter.

coordinate space (optional)

Possible values are STAGE and USER. See `:move:absolute?` for a description of coordinates spaces. The [*theta*] parameter is required to specify this parameter.

NOTE

On the auxiliary chuck, `:mov:abs and :mov:abs?` *use different coordinate spaces. For example, the command* `":move:abs? 2 stage"` *returns:* `"89623 -85101"` *but* `":move:abs 2 89623 -85101"` *is not the same location. If you need to move around on the auxiliary chuck, use relative moves.*

The commands uses the same coordinate spaces on the wafer chuck.

Example

Lowers the chuck, moves to the specified user coordinates, and raises the chuck again. In this example, coordinates are in microns. The query form of the move command verifies the device position.

```
:mov:abs 2 11980 -34 5000
:mov:abs? 2
+011980 -000034 +5000
```

Related Commands

`:move:absolute`
`:move:absolute:chuck`
`:move:absolute:options`
`:move:relative`
`:set:unit`

:move:absolute?

Short Command

`:mov:abs?`

devID

coordinate space

[user] [theorywiththeta] [theory] [stage]

Description

This query returns the device's current position in user coordinates. The values returned are not affected by wafer alignment, the load angle, the wafer map angle, or the wafer map axis settings. If units are in microns, values are returned as six-digit numbers, with preceding 0s to make up the six digits. For example, if the current position is at x, y, and z coordinates 1, -524, and 5000, the query returns the following:

```
+000001 -000524 +005000
```

Otherwise, if units are in mils, returns are in floating-point format. Theta, in milli-degrees, is also returned in floating-point format.

The coordinates are relative to the location currently defined as the user origin. You can use the `:set:preset` command to change the origin of the coordinate system.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

coordinate space

Possible values for coordinate space are: STAGE, USER, THEORYWITHTHETA, and THETA.

USER – User coordinates are user defined coordinates for X and Y that take into account user defined zero positions, angle, thermal compensation, and corrected stage values from the .cal file. The Z axis only applies compensation for the .cal file, not user defined zero.

STAGE – coordinates are absolute positions that do NOT take into account user defined zero, angle, or thermal compensation. They do contain corrections from the .cal file.

THEORYWITHTHETA – Same as STAGE, but theta position is returned with response. The theta parameter is padded with blank space characters. If theta is .29 then five spaces will be inserted. If theta is 123.0 then three spaces will be inserted.

THEORY – Same as STAGE.

NOTE

X, Y, Z and Theta are returned in integer formats if units are in metric, or in floating-point if in mils. If the units are in metric_float,s then floating point accuracy for metric units is returned. In addition if using the chuck device ID theta is returned.

NOTE

On the auxiliary chuck, `:mov:abs and :mov:abs?` *use different coordinate spaces. For example, the command "*`:move:abs? 2 stage`*" returns: "*`89623 -85101`*" but "*`:move:abs 2 89623 -85101`*" is not the same location. If you need to move around on the auxiliary chuck, use relative moves.*

The commands uses the same coordinate spaces on the wafer chuck.

Example

Lowers the chuck, moves to the specified user coordinates and raises the chuck again. You then use the query form of the move command to verify the device position.

```
:mov:abs 2 +11980 -34 5200
:mov:abs? 2
+011980 -000034 +5200
```

Note that Stage, Theory, and TheoryWithTheta are all system coordinates.

```
:mov:abs:opt 2 001234 004567 005000 TheoryWithTheta on on 123
:mov:abs? 2 TheoryWithTheta
+001234 +004567 +005000    123.00
```

Related Commands

:move:absolute
:set:unit
:move:relative
:move:absolute:options
:move:absolute:chuck
:move:relative
:set:preset

:move:absolute:chuck

Short Command

`:mov:abs:chuck`

The `:move:absolute:chuck` and the `:move:absolute` commands are identical in function.

devID

x

[none]

y

[none]

z

[up]
[down]
[none]

theta

[none]

coordinate space

USER
STAGE

Description

This command moves the device to the specified coordinates. The probe station moves the device's z-axis to a safe position before moving the device in x and y. Then it moves the z-axis to the supplied position after the device reaches the new location. The destination is not affected by wafer alignment, the load angle, the wafer map angle, or the wafer map axis settings.

The coordinates specified as USER are relative to the location currently defined as the user origin. You can use the `:set:preset` command to change the origin of the coordinate system.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

x

Specifies the x-axis coordinate to which the device will move when this command is executed. This number can be an integer, floating-point, or the character string none. If none, there is no change in the x-axis position.

The actual limits for the value of this parameter depend on the origin of the user coordinate system. The x coordinate value is interpreted as mils or microns. Use the `:set:unit` command to switch between mils and microns.

y

Specifies the y-axis coordinate to which the device will move when this command is executed. The legal values for this parameter are the same as previously described for the x-axis coordinate.

z

Specifies the device height at the end of the move. This number can be an integer, floating-point, or the character strings: up, down, or none. Entering none for the z position will cause the z-axis to return to its present position after separation is ensured for safe x, y, or theta movement.

For probe stations with pneumatic chucks, use one of the following character strings: up, down, or none. If none, the chuck height remains unchanged.

For probe stations with motorized chucks, this parameter can be an integer, floating-point, or the character strings up, down, and none.

theta (optional)

The theta parameter defines the final destination for the theta axis. It is only valid for the chuck channel of motion. Theta is in milli-degrees. This parameter is required to specify the [*coordinate space*] parameter.

coordinate space (optional)

Possible values are STAGE and USER. See `:move:absolute?` for a description of coordinates spaces. The [*theta*] parameter is required to specify this parameter.

Example

Lowers the chuck, moves the device to the specified user coordinates and raises the chuck again. In this example, coordinates are in microns. The query form of the move command verifies the device position.

```
:mov:abs:chuc 2 11980 -34 5000
:mov:abs? 2
+011980 -000034 +5000
```

Related Commands

`:move:absolute`
`:set:unit`
`:move:relative`
`:set:preset`
`:move:absolute:options`

:move:absolute:options

Short Command

`:mov:abs:opt`

devID

x

[loc], [none]

y

[loc], [none]

z

[loc], [none]

coordinate space

[user], [Theorywiththeta], [theory], [stage]

move Z

[on], [on], [loc]

wait

[off], [off]

theta

[loc]

Description

This command allows the user to specify the type of coordinates being used in specifying the x-, y- and z-axis and theta positions.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

x

Specifies the x-axis user coordinate to which the device will move when this command is executed. This number can be an integer, or the character string none. If none, there is no change in the x-axis position.

The actual limits for the value of this parameter depend on the origin of the user coordinate system. The x coordinate value is interpreted as mils or microns. Use the `:set:unit` command to switch between mils and microns.

y

The coordinate value (in microns or mils) that the device will move along the y-axis, relative to the current device position. Refer to the x-parameter description for information about the valid range values for this coordinate.

z

Specifies the z-axis height at the end of the move relative to the current height. This number can be an integer, floating-point, or the character strings: **up, down,** and **none**. Entering **none** for the z position will cause the z-axis to return to its present position after separation is ensured for safe x, y, or theta movement.

For probe stations with pneumatic chucks, use one of the following character strings: **up, down,** or **none**. If **none**, the chuck height remains unchanged.

For probe stations with motorized chucks, this parameter can be an integer or the character strings **up, down,** and **none**.

coordinate space

x, y, and z are returned in integer formats if units are in microns, or in floating-point if in mils.

User coordinates are user defined coordinates that take into account user defined zero positions, angle, thermal compensation and corrected stage values form the .cal file.

TheoryWithTheta returns the theta parameter.

Theory and Stage coordinates use the system's coordinates.

moveZ

Either ON or OFF. If set to OFF, the z-axis will not move to separate before a horizontal move. If set to ON, the z-axis will lower to separate.

wait

Required for backwards compatibility. However, it’s implemented as ON, i.e., this command will not return until all moves have completed.

theta

Specifies theta-axis. This number can be an integer, floating-point, or the character string none. Theta coordinates are in milli-degrees.

Example

The first command tells the probe station to move the chuck to a safe (retracted) position. Next, the device is moved to the position specified by the x- and y-type parameters, and then the z-axis is moved. Because the wait is set ON (wait OFF is not implemented), this command will not return until all moves have completed. The second command queries the station for its position in stage coordinates.

```
:mov:abs:opt 2 001234 004567 005000 stage on on
:mov:abs? 2
001234 004567 005000
```

Related Commands

`:move:absolute`
`:move:contact`
`:set:unit`
`:move:absolute:chuck`
`:move:relative`
`:set:preset`

:move:align

Short Command

`:mov:align`

Description

This command moves the device to the alignment position. The device z-axis is moved to its retracted position prior to xy-axis movement.

You can use the `:move:contact`, `:move:separate`, and `:set:vacuum` commands with this command to control vacuum and chuck status.

Parameters

None

Example

Centers the device, then uses the `:move:align` command to move the the alignment position.

```
:mov:cent 2
:mov:alig
```

Related Commands

:move:load

:move:center

Short Command

`:mov:cent`
devID

Description

This command moves the device to the center (machine 0,0) of the probe station. The device z-axis is moved to its retracted position prior to xy-axis movement.

You can use the `:move:contact`, `:move:separate`, and `:set:vacuum` commands with this command to control vacuum and chuck status.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation.

Example

Centers the device, then uses the `:move:contact` command to raise the chuck.

```
:mov:cent 2
:mov:cont 2
```

Related Commands

:move:contact
:move:load

:move:contact

Short Command

`:mov:cont`
devID

Description

This command moves the z-axis to the wafer contact position. It does not turn vacuum on or off.

In general, you use this command after a device movement, when the z-axis was away from the wafer — for example, before the device moved. If the chuck is down and the probe station receives a move command, the probe station makes the device movement, but does not raise the chuck after the move is completed. The `:move:contact` command allows you to raise the chuck again.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs (all but microscope).

Example

Shows the contact command used with one of the move-probe plan commands. In this example, the probe station lowers the chuck, moves the device, then repositions the device at a subsite. Once the device is correctly located over the subsite, the probe station raises the chuck.

```
:mov:sep 2
:mov:prob:next
:mov:prob:abs:subs 2
:mov:cont 2
```

Shows the contact command used with the query to determine if the chuck was raised completely.

```
:mov:cont 2
:mov:cont? 2
TRUE
```

Related Commands

`:move:contact?`
`:move:separate`

:move:contact?

Short Command

`:mov:cont?`
devID

Description

This query returns the character string TRUE if the device is at contact position. Any position 10 microns below or above the set contact position will return TRUE, otherwise it returns FALSE.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs (all but microscope).

Example

Shows the contact command used with the query to determine if the chuck was raised completely.

```
:mov:cont 2
:mov:cont? 2
TRUE
```

Related Commands

`:move:contact`
`:move:separate`

:move:down

Short Command

`:mov:down`
devID

Description

This command moves the device to the preset down position. The device xy axis does not change. The device remains down after the move to a new location, or until another command is given. Use the command `:move:separate` to move to the separate position.

The command is the opposite of the `:move:up` command.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs (all but microscope).

Example

Moves the device to the down position This example assumes that the device was up initially.

```
:mov:down 2
```

The following example shows the down command used with the query to determine if the device is at the down position.

```
:mov:down 2
:mov:down? 2
TRUE
```

Related Commands

`:move:contact`
`:move:down?`
`:move:up?`
`:move:up`

:move:down?

Short Command

`:mov:down?`
`devID`

Description

This query returns the character string TRUE if the device is at the preset down position, otherwise it returns FALSE.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs (all but microscope).

Example

Shows that the device is at the down position.

```
:mov:down 2
:mov:down? 2
TRUE
```

Related Commands

`:move:up`
`:move:up?`
`:move:down`

:move:kill

Short Command

`:mov:kill`
devID

Description

Used to stop motion on the specified device. This command can be used to disable the auto hold correction of micropositioners.

Parameters

devID

Device ID of the channel to kill motion on.

Example

Stops all motion of the first positioner.

`:move:kill 7`

:move:load

Short Command

`:mov:load`

Description

This command moves the device to the load position.

The location of the load position depends on its preset value. Summit 12000-Series probe stations load in the front center.

Parameters

None

Example

Use the `:move:load` command to load a wafer on the chuck.

When the probe station receives the command, it drops the chuck (if it is not already down) and moves the device to the front of the station to the load position.

`:mov:load`

Related Commands

`:move:center`

:move:microscope:absolute

Short Command

`:mov:micr:abs`
position

Description

Moves the fine focus stage of the eVue microscope to an absolute position. The eVue microscope has a 2mm range. The absolute location is in microns. 0 is the bottom of travel and 2000 is the top. Larger absolute locations moves the focus stage away from the chuck.

Parameters

position

A value in microns between 0 and 2000.

Example

Moves the fine focus stage to the center of travel.

```
:move:microscope:abs 1000
```

:move:microscope:absolute?

Short Command

```
:mov:micr:abs?
```

Description

Returns the current stage location of the eVue fine focus stage. The absolute location is in microns. 0 is the bottom of travel and 2000 is the top.

Parameters

None

Example

Moves the fine focus stage to the center of travel, then query the location, which returns 1000.

```
:move:microscope:abs 1000
:move:microscope:abs?
1000
```

:move:microscope:focus

Short Command

```
:mov:micr:foc
```

Description

Performs an Auto-Focus operation. The full range of travel for the eVue microscope is 2mm. The Up Distance and Down Distance can be used together to form a search region. If the current focus stage position is 500 and the parameters are 1000, 1000 the Auto-Focus operation starts at 0 and moves up to 1500 as the top location.

Parameters

Up Distance

The distance in microns that the fine focus stage moves up during an Auto-Focus.

Down Distance

The distance in microns that the fine focus stage moves down during an Auto-Focus.

Example

Runs an auto focus with a search band of 1000 microns above and 1000 below the current location. The position of the focus stage at the end of the operation is the location that has the best focus.

```
:move:microscope:focus 1000 1000
:move:microscope:abs?
465
```

:move:probeplan:absolute:die

Short Command

```
:mov:prob:abs:die
```

x

y

Description

Move to an absolute die location on the wafer. The move will be adjusted for the alignment angle (small angle). In addition, the move is affected by the load and wafer map angles, and the wafer map axis settings.

There is no check to ensure that the target index position is marked for testing in the probe plan. The move takes place regardless, and it is not considered an error if the position is unmarked. The axis directions and origin relate only to the current probe plan. Using 0,0 for the destination will move the wafer to the reference die.

Before using the move commands, you should define test sites with the probe plan Editor menus.

Parameters

x

Specifies the x die index, defined in the probe plan.

y

Specifies the y die index, defined in the probe plan.

Example

Tells the probe station to move to an index location. You then use the query form of the command to verify the device position.

```
:mov:prob:abs:die 30 40
:mov:prob:abs:die?
30 40
```

Related Commands

:move:probeplan:absolute:die?
:move:probeplan:last:die
:move:probeplan:prior:die
:move:probeplan:first:die
:move:probeplan:next:die
:probeplan:load

:move:probeplan:absolute:die?

Short Command

:mov:prob:abs:die?

Description

Returns the current die location. When the stage is on the reference die, 0,0 will be returned. The values returned will map to the values in the :move:prob:abs:die command. This means the values are affected by the load angle, the wafer map angle and the wafer map axis settings.

Parameters

None

Example

Moves the device to an index location. You then use the query form of the command to verify the device position.

```
:mov:prob:abs:die 30 40
:mov:prob:abs:die?
30 40
```

Related Commands

:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:prior:die
:move:probeplan:absolute:die
:move:probeplan:first:die
:probeplan:load

:move:probeplan:absolute:index

Short Command

`:mov:prob:abs:ind`
n

Description

This command moves the stage to the *nth* die site listed in the active probe plan. This command returns an error if the specified site has not been defined, such as a test die.

Before using the move commands, you should define test sites with the probe plan Editor menus. The probe station lowers the chuck, moves to the specified location, and then raises the chuck again.

Parameters

n

The die-site index number indicates the position of the site in the probe plan die-site list. The index is a positive integer up to five digits long.

Example

Loads a probe plan file and move the device to the 8th die site listed in the probe plan. The probe station is then queried for the current probe position.

```
:prob:load 65ghz_t1.wfd
:mov:prob:abs:index 8
:mov:prob:abs:index?
8
```

After loading the probe plan file, the probe station prompts you to position the probes on the alignment die of your substrate. When the probes are aligned, you can use the move command to go to the site, as shown above.

Related Commands

:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:next:site
:move:probeplan:prior:die
:move:probeplan:prior:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:subsite
:move:probeplan:prior:site
:move:probeplan:absolute:index?

:move:probeplan:absolute:index?

Short Command

`:mov:prob:abs:ind?`

Description

This query returns the current die site number that is under the probes. If the probes are not over a die site that is selected for testing, then an error condition is returned.

Parameters

None

Example

The following commands load a probe plan file and move the device to the 8th die site listed in the probe plan. The probe station is then queried for the current probe position.

```
:prob:load 65ghz_t1.wfd
:prob:alig
:mov:prob:abs:ind 8
:mov:prob:abs:ind?
8
```

After loading the probe plan file, the probe station prompts you to position the probes on the alignment die of your substrate. When the probes are aligned, you can use the move command to go to the site, as shown above.

Related Commands

`:move:probeplan:absolute:index`

:move:probeplan:absolute:index:label

Short Command

`:mov:prob:abs:ind:lab`
Die index
Subsite label

Description

This command will move to a die/subsite location. The die parameter is defined as the Nth testable die and the subsite is defined by the label. The label can include spaces. If spaces are included, the label must be enclosed in quotes.

Parameters

Die index
Testable die index of the currently loaded wafer map.

Subsite label
Label of the subsite for the move.

Example

Moves to the subsite labeled “voltage” of the 11th testable die.

```
:mov:prob:abs:ind:lab 11 voltage
:mov:prob:abs:ind:lab 11 “voltage out”
```

:move:probeplan:absolute:index:subsite

Short Command

`:mov:prob:abs:ind:subs`
Die index
Subsite label

Description

This command will move to a specified die and subsite, using the testable die index and the numeric index for the subsite.

Parameters

Die index
Testable die index of the currently loaded wafer map

Subsite label
Label of the subsite for the move

Example

Moves to the 5th subsite of the 11th testable die.

```
:mov:prob:abs:ind:subs 11 5
```

:move:probeplan:absolute:location

Short Command

`:mov:prob:abs:loc`
die x
die y
subsite

Description

This command moves the stage to a subsite location on a die on the wafer. The direction of the move is affected by the load angle, the wafer map angle and the wafer map axis settings (the same as the :move:prob:abs:die command).

The probe station lowers the chuck, moves to the specified location, and then raises the chuck again.

Parameters

die x
Specifies the *x* die index, defined in the probe plan.

die y
Specifies the *y* die index, defined in the probe plan.

subsite
The *n*th subsite of the specified die.

Example

Loads a probe plan file and then moves the device to the die index 3,2 subsite location 4.

```
:prob:load 65ghz_t1.wfd
:mov:prob:abs:loc 30 40 4
```

Related Commands

:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die
:move:probeplan:first:subsite
:move:probeplan:next:site
:move:probeplan:prior:die
:move:probeplan:prior:subsite
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:subsite
:move:probeplan:prior:site
:move:probeplan:absolute:index?

:move:probeplan:absolute:subsite

Short Command

`:mov:prob:abs:subs`
n

Description

This command moves the stage to the specified subsite on the current die. The subsite itself is an offset of x- and y-axis coordinates from the current die site listed in the active probe plan file.

The probe station lowers the chuck, moves to the specified location, and then raises the chuck.

Parameters

n

The die-site index number indicates the position of the site in the probe plan die-site list. The index is a positive integer up to five digits long.

Example

The `:move:probeplan:absolute:subsite` command is usually used with the `:move:probeplan:next:die` and other die-movement commands.

This example assumes that the chuck is up before device movements are made. The following commands load a probe plan, move to a die, move to a subsite on the die.

```
:prob:load 65ghz_t1.wfd
:prob:alig
```

After you load the probe plan file, the probe station asks you to position the probes on the alignment die of your substrate. When the probes are aligned, you can use the move-to-first-die and subsite commands, as shown.

```
:mov:prob:firs:die
:mov:prob:abs:subs 1
```

After performing your testing tasks, you can move to subsequent subsites and make additional tests, as shown with these commands:

NOTE

Use :move:next:site to simplify this example.

```
:mov:prob:next:die
:mov:prob:abs:subs 1
:mov:prob:abs:subs 2
:mov:prob:next:die
:mov:prob:abs:subs 2
:mov:prob:abs:subs 1
```

Related Commands

:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:next:site
:move:probeplan:prior:die
:move:probeplan:prior:subsite
:move:probeplan:absolute:index?
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:subsite
:move:probeplan:prior:site
:move:probeplan:absolute:index

:move:probeplan:absolute:subsite?

Short Command

`:mov:prob:abs:subs?`

Description

This query returns the subsite index that is currently under the probes. If the probes are not over a defined subsite position, an error code will be returned.

Parameters

None

Example

Uses the query to confirm the subsite index number.

```
:mov:prob:firs:die
:mov:prob:abs:subs 1
:mov:prob:abs:subs?
1
```

Related Commands

:move:probeplan:absolute:subsite

:move:probeplan:absolute:subsite:label

Short Command

`:mov:prob:abs:subs:lab`
Label

Description

This command will move to the subsite with the given label on the current die. The label can include spaces, and does not have to be enclosed in quotes.

Parameters

Label
Text string label of the desired subsite.

Example

Moves to the subsite labeled “voltage” of the current die.

```
:mov:prob:abs:subs:lab voltage
```

:move:probeplan:first:die

Short Command

```
:mov:prob:firs:die
```

Description

This command moves the stage to the first die site marked for testing in the active probe plan file. The probe station lowers the chuck, moves to the first die location, then raises the chuck.

Parameters

None

Error Condition

A probe plan must be loaded before the probe station can execute this command. This command also assumes that you have adjusted theta for your DUT. If you load a probe plan, but have not yet adjusted theta, this command moves the device to the location it assumes to be the first die marked for testing. If theta has not been properly adjusted, the new location can be inaccurate.

Example

Enter the commands that load a probe plan into the probe station and move to the first test die.

```
:prob:load 65ghz_t1.wfd
:prob:alig
:mov:prob:firs:die
```

The next command lines show that you can enter other commands, such as `:move:abs`, in between `:move:probeplan` commands. In the following example, you use the next-die command to return the device to the next die index listed in the active probe plan file.

```
:mov:abs 2 -181 4500  up
:mov:prob:next:die
:mov:prob:abs:subs 2
:mov:prob:first:die
```

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:next:site
:move:probeplan:prior:die
:move:probeplan:prior:subsite
:move:probeplan:absolute:index?
:move:probeplan:absolute:die
:move:probeplan:last:die
:move:probeplan:next:subsite
:move:probeplan:prior:site
:move:probeplan:absolute:index

:move:probeplan:first:subsite

Short Command

`:mov:prob:firs:subs`

Description

This command moves the stage to the first subsite of the current die designated for testing in the active probe plan file. The subsite itself is an offset of x- and y-axis coordinates from the current die site.

The probe station lowers the chuck, moves to the specified location, and then raises the chuck.

Parameters

None

Error Condition

This command returns an error if the specified subsite has not been defined.

Example

The `:move:probeplan:first:subsite` command is usually used with the `:move:probeplan:next:die` and other die-movement commands.

This example assumes that the chuck is up before device movements are made. The following commands load a probe plan, move to the first die, move to the first subsite on the die, then move to another subsite.

```
:prob:load 65ghz_t1.wfd
:prob:alig
```

After you load the probe plan file, the probe station asks you to position the probes on the alignment die of your substrate. When the probes are aligned, you can use the move-to-first-die and subsite commands, as shown.

```
:mov:prob:firs:die
:mov:prob:first:subs
```

After performing your testing tasks, you can move to the next subsite and make another test, as shown with these commands:

```
:mov:prob:abs:subs 2
```

Related Commands

`:move:probeplan:absolute:subsite`
`:move:probeplan:absolute:subsite?`
`:move:probeplan:absolute:die?`
`:move:probeplan:next:site`
`:move:probeplan:prior:die`
`:move:probeplan:first:die`
`:move:probeplan:last:die`
`:move:probeplan:prior:subsite`
`:move:probeplan:absolute:index?`
`:move:probeplan:absolute:die`
`:move:probeplan:last:die`
`:move:probeplan:next:subsite`
`:move:probeplan:prior:site`
`:move:probeplan:absolute:index`

:move:probeplan:last:die

Short Command

`:mov:prob:last:die`

Description

This command moves the stage to the last die site designated for testing in the active probe plan file.

The probe station lowers the chuck, moves to the last die location, then raises the chuck.

Parameters

None

Example

The following command lines show the device moving from one die to the next. Then to the last die.

```
:mov:prob:next:die
:mov:prob:next:die
:mov:prob:last:die
```

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:next:site
:move:probeplan:prior:die
:move:probeplan:prior:subsite
:move:probeplan:absolute:index?
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:next:subsite
:move:probeplan:prior:site
:move:probeplan:absolute:index

:move:probeplan:last:subsite

Short Command

```
:mov:prob:last:subs
```

Description

This command moves the stage to the last subsite of the current die listed in the active probe plan file.
The probe station lowers the chuck, moves to the last subsite location, then raises the chuck.

Parameters

None

Example

The following command lines show the device moving from one subsite to the next, then to the last subsite marked for testing.

```
:mov:prob:next:die
:mov:prob:next:subs
:mov:prob:next:subs
:mov:prob:next:subs
:mov:prob:last:subs
```

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:subsite
:move:probeplan:prior:site
:move:probeplan:absolute:index
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:next:site
:move:probeplan:prior:site
:move:probeplan:absolute:index?

:move:probeplan:next:die

Short Command

`:mov:prob:next:die`

Description

This command moves the stage to the next die site listed in the active probe plan file.
The probe station lowers the chuck, moves to the specified location, and then raises the chuck.

Parameters

None

Error Condition

This command returns an error if the device is already positioned at the last die to be tested because there is no next die.

Example

Positions the device at the next die site listed in the active probe plan file:

```
:mov:prob:next:die
```

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:site
:move:probeplan:prior:die
:move:probeplan:prior:subsite
:move:probeplan:absolute:index?
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:last:subsite
:move:probeplan:next:subsite
:move:probeplan:prior:site
:move:probeplan:absolute:index

:move:probeplan:next:site

Short Command

`:mov:prob:next:site`

Description

This command moves the stage to the next die site or subsite (whichever is next) listed in the active probe plan file.
The probe station lowers the chuck, moves to the specified location, and then raises the chuck.

Parameters

None

Error Condition

This command returns an error if the device is already positioned at the last site to be tested because there is no next site.

Example

Positions the device at the next test site listed in the active probe plan file:

`:mov:prob:next:site`

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:prior:die
:move:probeplan:prior:subsite
:move:probeplan:absolute:index?
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:last:subsite
:move:probeplan:next:subsite
:move:probeplan:prior:site
:move:probeplan:absolute:index

:move:probeplan:next:subsite

Short Command

`:mov:prob:next:subs`

Description

This command moves the stage to the next subsite of the current die listed in the active probe plan file.
The probe station lowers the chuck, moves to the specified location, and then raises the chuck.

Parameters

None

Error Condition

This command returns an error if there are no defined subsites or if the device is already positioned at the last subsite to be tested because there is no next subsite.

Example

Shows how to tell the probe station to position the device at the next subsite listed in the active probe plan file:

`:mov:prob:next:subs`

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:prior:die
:move:probeplan:prior:subsite
:move:probeplan:absolute:index?
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:last:subsite
:move:probeplan:next:site
:move:probeplan:prior:site
:move:probeplan:absolute:index

:move:probeplan:prior:die

Short Command

`:mov:prob:pri:die`

Description

This command moves the stage to the previous (prior) die site listed in the active probe plan file. The probe station lowers the chuck, moves to the specified location, and then raises the chuck.

Parameters

None

Error Condition

This command returns an error if the probes are already located on the first die because there is no prior die to move to.

Example

The following command line moves the device to the prior die site listed in the currently active probe plan file:

```
:mov:prob:pri:die
```

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:next:subsite
:move:probeplan:prior:subsite
:move:probeplan:absolute:index?
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:last:subsite
:move:probeplan:next:site
:move:probeplan:prior:site
:move:probeplan:absolute:index

:move:probeplan:prior:site

Short Command

`:mov:prob:pri:site`

Description

This command moves the stage to the previous (prior) die site, or subsite, listed in the active probe plan file. If the probes are located at the first subsite in the list, the device moves to the prior die site. Otherwise, the device moves to the prior subsite.

The probe station lowers the chuck, moves to the specified location, and then raises the chuck.

Parameters

None

Error Condition

This command returns an error if the probes are already located at the first subsite on the first die because there is no prior site to move to.

Example

The following command line moves the device to the prior site listed in the currently active probe plan file.

`:mov:prob:pri:site`

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:next:subsite
:move:probeplan:prior:subsite
:move:probeplan:absolute:index?
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:last:subsite
:move:probeplan:next:site
:move:probeplan:prior:site
:move:probeplan:absolute:index

:move:probeplan:prior:subsite

Short Command

`:mov:prob:pri:subs`

Description

This command moves the stage to the previous (prior) subsite of the current die listed in the active probe plan file. The probe station lowers the chuck, moves to the specified location, and then raises the chuck.

Parameters

None

Error Condition

This command returns an error if the probes are already located on the first subsite listed in the subsite list of the current die, because there is no prior subsite.

Example

The following command line moves the device to the prior subsite listed in the currently active probe plan file.

`:mov:prob:pri:subs`

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:next:subsite
:move:probeplan:prior:site
:move:probeplan:absolute:index?
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:last:subsite
:move:probeplan:next:site
:move:probeplan:prior:die
:move:probeplan:absolute:index

:move:probeplan:relative:die

Short Command

`:move:prob:rel:die`

xIndex

[none]

yIndex[none]

[none]

z

[none]
[up]
[down]

Description

This command allows you to move to a die location relative to the current location. The direction of the move is affected by the load angle, the wafer map angle and the wafer map axis settings.

Moves can only be made to a defined die site on the map. Also, you must start the move from a defined die site. Attempts to start from or move to non-die locations will result in an error. Positive x and y values defined when the wafer map was created determine the actual direction of travel. This command is the same as: `:move:probeplan:relative:index`.

Parameters

xIndex

Indicates the new die location relative to the current die. Valid entries are integer or none.

yIndex

Indicates the new die location relative to the current die. Valid entries are integer or none.

z

Determines the position of the z-axis after travel has completed. Valid entries are up, down, or none, and the actual position (Summit 12000-Series stations only). Accepts floating-point arguments.

Example

A series of relative moves will return to the original die with the chuck in the down position. Note that before a relative move can be issued, a probe plan must be loaded, aligned and the stage moved to a defined die position.

```
:prob:load "\CMI\300m_fet.wfd"
:prob:alig
:mov:prob:abs:index 80
:mov:prob:rel:index 2 0 UP
:mov:prob:rel:index 0 -3 UP
:mov:prob:rel:index -2 0 UP
:mov:prob:rel:index 0 3 DOWN
```

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:next:subsite
:move:probeplan:prior:site
:move:probeplan:absolute:index?
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:last:subsite
:move:probeplan:next:site
:move:probeplan:prior:die
:move:probeplan:absolute:index

:move:probeplan:relative:index

Short Command

`:move:prob:rel:ind`

xIndex

[none]

yIndex[none]

[none]

z

[none]
[up]
[down]

Description

This command allows you to specify relative moves within the active probe plan wafer map. Moves can only be made to a defined die site on the map. Also, you must start the move from a defined die site. Attempts to start from or move to non-die locations will result in an error. Positive x and y values defined when the wafer map was created determine the actual direction of travel. This command is the same as: `move:probeplan:relative:die`.

Parameters

xIndex

Indicates the new die location relative to the current die. Valid entries are integer or none.

yIndex

Indicates the new die location relative to the current die. Valid entries are integer or none.

z

Determines the position of the z-axis after travel has completed. Valid entries are up, down, or none, and the actual position (Summit 12000-Series stations only). Accepts floating-point arguments.

Example

Shows a series of relative moves that will return to the original die with the chuck in the down position. Note that before a relative move can be issued, a probe plan must be loaded, aligned and the stage moved to a defined die position.

```
:prob:load "\CMI\300m_fet.wfd"
:prob:alig
:mov:prob:abs:index 80
:mov:prob:rel:index 2 0 UP
:mov:prob:rel:index 0 -3 UP
:mov:prob:rel:index -2 0 UP
:mov:prob:rel:index 0 3 DOWN
```

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:next:subsite
:move:probeplan:prior:site
:move:probeplan:absolute:index?
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:last:subsite
:move:probeplan:next:site
:move:probeplan:prior:die
:move:probeplan:absolute:index

:move:relative

Short Command

`:mov:rel`
devID
x
[none]
y
[none]
z
[none]
[up]
[down]
theta
[none]
[loc]

Description

This command moves the device z-axis to a safe position, moves to the specified x-, and y-axis positions (theta is optional), and then moves the z-axis to the specified position. Moves are made relative to the current position. The commanded move is not affected by the wafer alignment, the load angle, the wafer map angle, or the wafer map axis settings. The probe station waits for the device movement to complete before beginning execution of the next command.

This command is identical to the `:move:relative:chuck` command.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

x

The coordinate value (in microns or mils) that the device will move along the x-axis, relative to the current device position. This number can be an integer, floating-point, or the character string none. If none is entered there is no change in the x-axis direction.

The actual range of valid coordinate values depends on the current device position. If you enter a value outside the probe station limit switches, the probe station returns an error.

y

The coordinate value (in microns or mils) that the device will move along the y-axis, relative to the current device position. Refer to the x-parameter description for information about the valid range values for this coordinate.

z

Specifies the z-axis height at the end of the move relative to the current height. This number can be an integer, floating-point, or the character strings: up, down, or none. Entering none for the z position will cause the z-axis to return to its present position after separation is ensured for safe x, y, or theta movement.

If the Z parameter is “0” it is an explicit instruction not to change the Z height at the destination. The probe station auto separates if necessary and moves to the new location, but if Edge Sense is enabled it will not invoke a search for the sensor. Use the “UP” to invoke the search for Edge Sense.

For probe stations with a pneumatic chuck, use one of the following character strings: up, down, and none. If none, the chuck height remains unchanged.

For probe stations with a motorized chuck, this parameter can be an integer or the character strings up, down, and none.

theta (optional)

Specifies theta-axis. This number can be an integer, floating-point, or the character string none. Theta coordinates are in milli-degrees.

Example

The first command returns the current device position. The second command moves the device to a new position, and, when the move completes, returns the new position coordinates.

```
:mov:abs? 2
022659 -008430 000500
:mov:rel 2
604 181 100
:mov:abs? 2
022055 -008249 000600
```

Related Commands

`:move:absolute`
`:move:relative:chuck`
`:move:absolute`
`:move:absolute:options`

:move:relative:chuck

Short Command

`:mov:rel:chuc`
devID
x
[none]
y
[none]
z
[none]
[up]
[down]
theta
[none]
[loc]

Description

This command moves the device z-axis to a safe position, moves to the specified x-, and y-axis positions (theta is optional), and then moves the z-axis to the specified position. Moves are made relative to the current position. The probe station waits for the device movement to complete before beginning execution of the next command.

This command is identical to the `:move:relative` command.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

x

The coordinate value (in microns or mils) that the device will move along the x-axis, relative to the current device position. This number can be an integer, floating-point, or the character string none. If none is entered there is no change in the x-axis direction.

The actual range of valid coordinate values depends on the current device position. If you enter a value outside the probe station limit switches, the probe station returns an error.

y

The coordinate value (in microns or mils) that the device will move along the y-axis, relative to the current device position. Refer to the x-parameter description for information about the valid range values for this coordinate.

z

Specifies the z-axis height at the end of the move relative to the current height. This number can be an integer, floating-point, or the character strings: up, down, and none. Entering none for the z position will cause the z-axis to return to its present position after separation is ensured for safe x, y, or theta movement.

For probe stations with a pneumatic chuck, use one of the following character strings: up, down, and none. If none, the chuck height remains unchanged.

For probe stations with a motorized chuck, this parameter can be an integer or the character strings up, down, and none.

Specifies theta-axis. This number can be an integer, floating-point, or the character string none. Theta coordinates are in milli-degrees.

theta (optional)

Specifies theta-axis. This number can be an integer, floating-point, or the character string none. Theta coordinates are in milli-degrees.

Example

The first command returns the current device position. The second command moves the device to a new position, and, when the move completes, returns the new position coordinates.

```
:mov:abs? 2
022659 -008430 000500
:mov:rel:chuc
2 604 181 100
:mov:abs? 2
022055 -008249 000600
```

Related Commands

:move:absolute
:move:relative
:move:absolute
:move:absolute:options

:move:separate

Short Command

`:mov:sep`
devID

Description

This command moves the device to break the contact between the probes and the DUT.

Parameters

devID
2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

Example

Uses the separate command to lower the chuck so that the probes are no longer in contact with the wafer (DUT). The query is then used to determine if the chuck was lowered.

```
:mov:sep 2
:mov:sep? 2
TRUE
```

Related Commands

:move:contact
:move:separate

:move:separate?

Short Command

`:mov:sep?`
devID

Description

This query returns the character string TRUE if the device is 10 microns above or anywhere below the set separate position, otherwise it returns FALSE.

Parameters

devID
2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

Example

Uses the separate command to lower the chuck so that the probes are no longer in contact with the wafer. The query is then used to determine if the chuck was lowered.

```
:mov:sep 2
:mov:sep? 2
TRUE
```

Related Commands

:move:contact
:move:separate

:move:up

Short Command

`:mov:up`
devID

Description

This command is the same as the `:move:contact` command.

You can use the `:move:contact`, `:move:separate`, `:move:center`, and `:set:vacuum` commands with this command to control vacuum and chuck status.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

Example

Raises the chuck. This example assumes that the chuck was down initially.

```
:mov:up 2
```

The following example shows the query form used to determine if the chuck was raised completely.

```
:mov:up 2
:mov:up? 2
TRUE
```

Related Commands

:move:center
:move:down
:move:load
:move:contact
:move:up

:move:up?

Short Command

`:mov:up?`
devID

Description

This query returns the character string TRUE if the chuck position is equal to or above the set contact position, otherwise it returns FALSE.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

Example

Shows that the chuck is completely up.

```
:mov:up 2
:mov:up? 2
TRUE
```

Related Commands

:move:down
:move:up

Wafer Map Commands

Commands		
:probeplan:absolute:reference	:probeplan:nsubsites?	:probeplan:subsites:delete
:probeplan:align	:probeplan:ntested?	:probeplan:subsites:delete:all
:probeplan:die:location?	:probeplan:origin	:probeplan:subsites:label?
:probeplan:die:size	:probeplan:origin?	:probeplan:subsites:total?
:probeplan:die:xsize?	:probeplan:quality:size	:probeplan:subsites:xoffset?
:probeplan:die:ysize?	:probeplan:quality:size?	:probeplan:subsites:yoffset?
:probeplan:flatnotch	:probeplan:reference	:probeplan:test:die
:probeplan:flatnotch?	:probeplan:reference?	:probeplan:test:die?
:probeplan:grid:shift	:probeplan:save	:probeplan:testing:sequence
:probeplan:grid:shift?	:probeplan:set:palette	:probeplan:testing:sequence?
:probeplan:load	:probeplan:set:subsite	:probeplan:wafer:orientation
:probeplan:load?	:probeplan:set:subsite:label:active?	:probeplan:wafer:orientation?
:probeplan:mark:die	:probeplan:show	:probeplan:wafersize
:probeplan:mark:xydie	:probeplan:street:size	:probeplan:wafersize?
:probeplan:mark:xydie	:probeplan:street:xsize?	:probeplan:xy:orientation
	:probeplan:street:ysize?	:probeplan:xy:orientation?

:probeplan:absolute:reference

Short Command

```
:prob:abs:ref
```

Description

Moves the reference die to the specific X, Y location without moving the origin at the same time. It is different from `:prob:ref` which does move the origin when a new location is specified.

Parameters

X
Defines the column or X location in user coordinate space.

X
Defines the row or Y location in user coordinate space.

Example

Moves the reference location to the die 3, 4 using user coordinate space.

```
:prob:abs:ref 3 4
```

Related Commands

:probeplan:reference
:probeplan:reference?
:probeplan:origin
:probeplan:origin?

:probeplan:align

Short Command

`:prob:alig`
pause
[on]
[off]
xPos
ypos

Description

This command sets the alignment position for a probe plan. If only *devID* is specified in the command line, the station reads the alignment position from the active probe plan and moves the station to that position. A message will appear that prompts you to move the stage to the actual alignment position and to press ENTER. In so doing, you allow the station to compensate for differences between the actual DUT and the probe plan wafer map.

The parameters, pause, xPos and yPos allow you to remotely control the alignment position and prompting. They must be used together.

CAUTION

If you do not use this command and the wafer has been moved since the probe plan was created, `:move:probeplan` *commands may not position the device exactly where you need it to be. You must issue this command immediately after a* `:probeplan:load` *command.*

Parameters

pause

Controls whether or not the station will prompt after it has reached alignment position. On indicates that prompting is enabled. This parameter is required to specify the [xPos] and [yPos] parameter.

xPos

Specifies the alignment position in USER coordinates to go to. This new position overrides the values stored in the probe plan file. The [pause] parameter is required to specify this parameter.

ypos

Specifies the alignment position in USER coordinates to go to. This new position overrides the values stored in the probe plan file. The [pause] and [xPos] parameters are required to specify this parameter.

Example

Shows a probe plan being loaded and the alignment position being set:

```
:prob:load "CMI\300m_fet.ppd"
:prob:alig
```

Shows a probe plan being loaded and the alignment position being set:

```
:prob:load "CMI\300m_fet.ppd"
:prob:alig OFF 2300 -12000
```

Uses the current stage location as the reference die location.

```
:ProbePlan:Align OFF
```

Related Commands

:probeplan:load
:probeplan:load?

:probeplan:die:location?

Short Command

`:prob:die:loc?`
testable index

Description

Returns the column row in wafer map user coordinate space of the given testable index number.

Parameters

testable die index
Index number from 1 to total die that indicates which testable index location the user wants information on.

Example

Returns the column / row in wafer map user coordinate space. Whitespace character in between.

```
:prob:die:loc?
3 2
```

:probeplan:die:size

Short Command

`:prob:die:size`
x
y

Description

Use this command to set the die size of the currently loaded wafer map file. The x parameter represents the horizontal size and y represents the vertical size of the die. The units used for the x and y parameters can be set using the `:set:unit` command.

Parameters

x
Indicates the x die size of the wafer map in micron or mils depending on the current units.

y
Indicates the y die size of the wafer map in micron or mils depending on the current units.

Example

Sets the die size to 30000x20000 in the currently loaded wafer map. The current units are in microns.

```
:prob:die:size 30000 20000
```

:probeplan:die:xsize?

Short Command

`:prob:die:xsiz?`

Description

This query command returns the horizontal size of the die in the currently loaded wafer map. The value is returned using the current units (see *:set:unit* command).

Parameters

None

Example

Returns the horizontal size of the die in the currently loaded wafer map. The current units are in microns.

```
:prob:die:xsize?
```

:probeplan:die:ysize?

Short Command

```
:prob:die:ysiz?
```

Description

This query command returns the vertical size of the die in the currently loaded wafer map. The value is returned using the current units (see *:set:unit* command).

Parameters

None

Example

Returns the vertical size of the die in the currently loaded wafer map. The current units are in microns.

```
:prob:die:ysize?
20000
```

:probeplan:flatnotch

Short Command

```
:prob:flat
```

Type

Size

Description

This command will set the FLAT or NOTCH on a wafer map. It does not set the orientation of the FLAT or NOTCH. To set the orientation, use `:prob:waf:ori`.

Parameters

Type

Possible values are FLAT or NOTCH. If Type is FLAT then size is the length of the flat. If Type is NOTCH then size is the diameter of the notch.

Size

Uses mm or inches based off of the `:set:unit` command.

Example

Sets a notch on the wafer to 5mm.

```
:prob:flat NOTCH 5
```

:probeplan:flatnotch?

Short Command

`:prob:flat?`

Description

This command returns a string that contains two values: *Type and Size.* Type is either FLAT or NOTCH and the *Size* depends on the *Type*. If *Type is* FLAT, size is the length of the flat. If *Type* is NOTCH, size is the diameter of the notch.

Parameters

None

Example

Returns that the wafer has notch 5mm in diameter and another wafer that has flat of 60mm.

```
:prob:flat?
NOTCH 5
:prob:flat?
FLAT 60
```

:probeplan:grid:shift

Short Command

`:prob:grid:shif`
x
y

Description

This command will shift the die pattern on the wafer map by X, Y microns.

Parameters

x
Specifies in microns the distance to shift the wafer map horizontally.

y
Specifies in microns the distance to shift the wafer map vertically.

Example

The wafer map will be shifted 1000, 1000 microns.

```
prob:grid:shif 1000 1000
```

:probeplan:grid:shift?

Short Command

`:prob:grid:shif?`

Description

This command will return the current shift of the die pattern on the wafer map.

Parameters

None

Example

Shifts the die pattern on the wafer map by -1000 microns in X and Y.

```
:prob:grid:shif?
-1000 -1000
```

:probeplan:load

Short Command

`:prob:load`
pathname

Description

This command loads a probe plan file from disk into RAM and makes the file active. Once a probe plan has been loaded, you can access it from the menu system or with commands. Sending this command will clear out any currently loaded probe plan file. If you would like to save the current probe plan before loading another one, use the `:probeplan:save` command.

Immediately following a `:probeplan:load` command, you must issue a `:probeplan:align` command.

Parameters

pathname

Specifies the full path and filename of the probe plan file you want to load. File and path names follow the Windows naming conventions. The filename must have a three-character file extension. In general, probe plan files have a .wfd extension. If the filename contains spaces, you must surround the filename with quotes.

Error Condition

If you try to load a file that is not a probe plan, this command returns an error. An error is also returned if you try to load a file that does not exist or that is not located at the specified path.

Example

Loads the probe plan file *example\new_test.wfd*.

```
:prob:load "c:\cmi\example\new_test.wfd"
```

Related Commands

:probeplan:align
:move:probeplan:absolute:die?
:move:probeplan:absolute:index?
:move:probeplan:absolute:subsite?
:move:probeplan:first:subsite
:move:probeplan:last:subsite
:move:probeplan:next:site
:move:probeplan:prior:die
:move:probeplan:absolute:die
:move:probeplan:absolute:index
:move:probeplan:absolute:subsite
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:prior:site
:probeplan:load?

:probeplan:load?

Short Command

`:prob:load?`

Description

This query returns the current probe plan filename. If no probe plan is loaded, an error string is returned.

Parameters

None

Example

Queries for the currently active probe plan filename.

```
:probeplan:load?
TEST_01.wfd
```

Related Commands

:probeplan:align
:move:probeplan:absolute:die?
:move:probeplan:absolute:subsite?
:move:probeplan:first:subsite
:move:probeplan:last:subsite
:move:probeplan:next:site
:move:probeplan:prior:die
:move:probeplan:prior:subsite
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:next:subsite
:move:probeplan:prior:site
:probeplan:load?

:probeplan:mark:die

Short Command

```
:prob:mark:die
```

n
color

Description

This command specifies the color used to identify die sites when you display a wafer map. The command uses indexes that are stored in the palette (see *:probeplan:set:palette* command).

Parameters

n

Indicates which of the tested die should be marked. The die number is the index number for the die marked as testable in the wafer map.

color

Specifies which color the wafer map uses to mark tested die. Use the index number corresponding to the color you want in the palette. There are 256 possible colors numbered 0 to 255.

Example

Sets the 10th testable die to the palette color that is stored in index 16.

```
:prob:mark:subs:val 10 4 3 45.6787
```

:probeplan:mark:xydie

Short Command

```
:prob:mark:xyd
```

x
y
color

Description

This command is very similar to the `:probeplan:mark:die` command, the only difference is that X,Y coordinates are used to specify the die location. Any die can be selected whether it is marked as testable or not. The command uses indices that are stored in the palette.

Parameters

x

X represents the die column for the specified die location defined in user coordinate space.

y

Y represents the die row for the specified die location defined in user coordinate space.

color

Specifies which color the wafer map uses to mark tested die. Use the index number corresponding to the color you want in the palette. There are 256 possible colors numbered 0 to 255.

Example

Sets the die at location 4, 5 to the palette color that is stored in index 16.

```
:prob:mark:xydie 4 5 16
```

:probeplan:nsubsites?

Short Command

```
:prob:nsub?
```

Description

This query returns the number of active subsites defined in the probe plan subsite list.

Parameters

None

Example

Requests the number of active subsites. In this case, there are eight subsites in the subsite list.

```
:prob:nsub?
8
```

Related Commands

`:move:probeplan:absolute:subsite`
`:probeplan:ntested?`

:probeplan:ntested?

Short Command

```
:prob:ntes?
```

Description

This query returns the number of die marked for testing in the currently active probe plan. If you have not yet loaded a probe plan file, this query returns nothing.

Parameters

None

Example

Requests the number of die marked for testing. In this example, you first load a probe plan file. You then enter the die-tested query. In this example, the number of die currently marked for testing is 181 die.

```
:prob:load "example\vband\65ghz_t1.wfd"
:prob:ntes?
18
```

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:prior:die
:move:probeplan:next:site
:move:probeplan:prior:site
:move:probeplan:first:subsite
:move:probeplan:next:subsite
:move:probeplan:prior:subsite

:probeplan:origin

Short Command

`:prob:orig`
X
Y

Description

This command can be used to set the origin for the currently loaded wafer map. The X and Y parameters are the column and row from the left bottom corner of the wafer map. Columns start at 0 on the left and are positive to the right. Rows start at 0 on the bottom and are positive to the top. The origin command is used to define the user coordinate space, so the X and Y parameters are in terms of the left bottom corner. Once the origin of a wafer map has been set up, all other commands that use column and row wafer coordinates will use the origin as a base. The origin is not visible on the wafer map. The X and Y die location can be any number. For example:

`:prob:orig -1 -1` will set the left, bottom corner to have coordinates 1,1.

Parameters

X
Defines the X column of the origin from the leftmost column.

Y
Defines the Y row of the origin from the bottommost row.

Example

Sets the origin 3 columns over from the left and 4 rows up from the bottom.

```
:prob:orig 3 4
:prob:orig?
3 4
```

:probeplan:origin?

Short Command

`:prob:orig?`

Description

This command queries the currently loaded wafer map for the origin die. The values returned are in relation to the left bottom corner of the wafer map and are in column, row (X, Y) order.

Parameters

None

Example

Shows that the origin is 4 columns over and 5 rows up from the left bottom corner of the wafer map.

```
:prob:orig?
4 5
```

:probeplan:quality:size

Short Command

`:prob:qual:size`
Size

Description

This command will set the quality area of a wafer. The quality area can not be larger than wafer size. The `:set:unit` command can be used to change the interpretation of size parameter. It can be in either English or Metric units.

Parameters

Size

Size of the quality area in mm or inches based on `:set:unit`.

Example

Sets the quality area of the wafer to 190mm.

```
:prob:qual:size 190
```

:probeplan:quality:size?

Short Command

`:prob:qual:size?`

Description

This command returns the quality area size of the current wafer map. It uses mm or inches based on `:set:unit` command.

Parameters

None

Example

Returns 190mm as the current quality area.

```
:prob:qual:size?
190
```

:probeplan:reference

Short Command

`:prob:ref`
X
Y

Description

This command can be used to set the reference and origin die for the currently loaded wafer map. The x- and y-parameters are the column and row in wafer map user coordinates. The reference is the die location of the probe to pad alignment. It appears as a red rectangle around the home die on the wafer map.

This command will also move the origin so the reference die will maintain the same user coordinates it had before the `:prob:ref` call. If you wish to give the reference die coordinates of x, y, make a "`:probeplan:origin 0 0`" call after `:prob:ref`. This combination will set the origin to the left-bottom with coordinates 0,0 and put the reference die at x, y.

Parameters

X

Defines the X column of the reference in user coordinates.

Y

Defines the Y row of the reference in user coordinates.

:probeplan:reference?

Short Command

`:prob:ref?`

Description

This command can be used to query the reference die for the currently loaded wafer map. The reference is the die location of the probe to pad alignment. It appears as an R on the wafer map. The return values are in wafer map user coordinates.

Parameters

None

Example

Returns the reference die of 3, 4.

```
:prob:ref?
3 4
```

:probeplan:save

Short Command

`:prob:save`
pathname

Description

This command saves the currently loaded wafer map file from memory onto disk. The pathname parameter is optional. If it is not supplied then the wafer map is saved using the current name and location.

Parameters

pathname

The fully qualified pathname to which the wafer map file will be saved. If a pathname is not specified, then the wafer map will be saved using the default name that it was opened with. If the filename contains spaces, you must surround the filename with quotes (see example).

Example

Shows how to save a wafer map file.

```
:prob:save "c:\cmi\example\new test.wfd"
```

:probeplan:set:palette

Short Command

`:prob:set:pal`
n
red
green
blue

Description

Use this command to define the colors in the palette. The relative intensity of red, green, and blue determines the color. Intensity ranges from 0 to 255, with 0 being off and 255 being the brightest intensity. There are 256 possible colors in the palette, numbered 0 to 255. The first 16 are predefined, as shown below. You can assign custom colors to any of the palette indexes.

Index – Color
0 – Black
1 – Blue
2 – Green
3 – Cyan
4 – Red
5 – Magenta
6 – Brown
7 – Dull White
8 – Dark Gray
9 – Light blue
10 – Light Green
11 – Light Cyan
12 – Light Red
13 – Light Magenta
14 – Light Yellow

Parameters

n
Use this parameter to specify the index number (that is, location in the palette) of the color you are defining. You can use any number between 0 and 255.

red
Sets the intensity of red in the pixels for the maximum bin. The intensity scale runs from 0 to 255, with 0 being no red and 255 being the brightest red.

green
Sets the intensity of green in the pixels for the maximum bin. The intensity scale runs from 0 to 255, with 0 being no green and 255 being the brightest green.

blue
Sets the intensity of blue in the pixels for the maximum bin. The intensity scale runs from 0 to 255, with 0 being no blue and 255 being the brightest blue.

Example

Defines color 100 to be red, then sets 100 as the color to display die #30 in the wafer map.

```
:prob:set:palette 100 255 0 0
:prob:mark:die 30 100
```

Related Commands

`:probeplan:mark:die`

:probeplan:set:subsite

Short Command

`:prob:set:subs`
index
x
y
{label}
{active}
[on]
[off]

Description

This command defines a subsite location (device position relative to the die site). The subsite is defined as x- and y-axis offsets from the die location. Optional parameters to this command are the label, and whether or not the subsite is active.

After defining a subsite, use the `:move:probeplan:absolute:subsite` command to position the device at the subsite location.

Parameters

index

Identifies the specific subsite. You may define up to 255 different subsites. Each subsite must have an index number to differentiate it from the other subsites. For example, if you have three subsites, use 0, 1, and 2 as the three subsite index numbers. The 0 subsite is also the reference point inside the wafer. If an index number is larger than the total number of subsites, the index will be changed to one greater than the last subsite.

x

The x offset for the subsite. This coordinate value is an offset from the defined die sites in the active wafer map. The values you can assign to this parameter depend on the size of the die.

y

The y offset for the subsite. This coordinate value is an offset from the defined die sites in the active wafer map file. The values you can assign to this parameter depend on the size of the die.

label

A label for this subsite. If it includes spaces, it must be enclosed in quotation marks. If not given, the default “Label” is assigned. This parameter is required to specify the [*active*] parameter.

active

Active defines whether or not this subsite is a part of the active subsite stepping sequence. See the *:move:probeplan:next:subsite* command. Possible values are “on” and “off”. If Active is not given then the subsite is “on” by default. The [*label*] parameter is required to specify this parameter.

Example

Defines subsite number 1 as being 24 microns in X and –117 microns in Y, away from the reference location. The label is given the value Test one and it is defined as being active.

```
:prob:set:subs 1 24 -117 “Test One” on
```

:probeplan:set:subsite:label:active?

Short Command

`:prob:set:subs:lab:act?`
subsite index

Description

This command returns the test status of a subsite at the given index. Possible return values are “ON” or “OFF”. If a subsite is “ON” then it is in the subsite stepping sequence. A subsite that is “OFF” is not in the stepping sequence.

Parameters

subsite index

Defines the subsite index number. Valid numbers range from 0 to 255.

Example

The subsite located at index 4 is not part of the subsite stepping sequence.

```
:prob:set:subs:lab:act? 4
OFF
```

:probeplan:show

Short Command

```
:prob:show
```

Description

This command controls the probe plan wafer map display.

Parameters

The [on] [off] parameter determines whether or not the wafer map is displayed.

[on] – Displays the map

The [on] [off] parameter determines whether or not the wafer map window is always displayed on top of other windows.

[on] – Always displays the map on top of other windows.

The [on] [off] parameter determines whether or not die site coordinate are displayed.

[on] – Displays the die site coordinates.

Example

Causes the wafer map to always be displayed on top of other windows, even when it is not the active window, and to display die site coordinates.

```
:prob:show on
```

:probeplan:street:size

Short Command

```
:prob:str:size
```

XSize

YSize

Description

This command sets the street size of the current wafer map in X and Y. The street size can not be larger than the die size. The size parameters may be in microns or mils.

Parameters

XSize

Size of the X Street in microns or mils.

YSize

Size of the Y Street in microns or mils.

Example

Sets the street size of the wafer map to 1000, 1000.

```
:prob:str:size 1000 1000
```

:probeplan:street:xsize?

Short Command

`:prob:str:xsiz?`

Description

This command returns the X street size of the current wafer map. It uses microns or mils based on `:set:unit`.

Parameters

None

Example

Returns that the size of the street is 1000 microns.

```
:prob:str:xsiz?
1000
```

:probeplan:street:ysize?

Short Command

`:prob:str:ysiz?`

Description

This command returns the Y street size of the current wafer map. It uses microns or mils based on `:set:unit`.

Parameters

None

Example

Returns that the size of the street is 1000 microns.

```
:prob:str:ysiz?
1000
```

:probeplan:subsites:delete

Short Command

`:prob:subs:del`
Index

Description

This command will delete the subsite at the specified index.

Parameters

Index
Specifies the subsite index to delete.

Example

Deletes the subsite located at index 4.

```
prob:subs:del 4
```

:probeplan:subsites:delete:all

Short Command

`:prob:subs:del:all`

Description

This command will delete all of the subsites in the subsite list.

Parameters

None

Example

Remove all subsites from the subsite list.

```
prob:subs:del:all
```

:probeplan:subsites:label?

Short Command

```
:prob:subs:lab?
```
subsite

Description

This command will return the descriptive text field used to identify the subsite. This label can be set with the `:probeplan:set:subsite` command.

Parameters

subsite

Defines the subsite index number. Valid numbers range from 0 to 255.

Example

Retrieves the descriptive label for the 3rd subsite, which is “Subsite three”.

```
:prob:subs:lab? 3
Subsite three
```

:probeplan:subsites:total?

Short Command

```
:prob:subs:tot?
```

Description

This command will return the total number of subsites whether they are marked as active or not. This command differs from the `:probplan:nsubsites?` in that the it returns the total number of subsites, not just the active ones.

Parameters

None

Example

Returns 15 as the total number of subsites defined in the wafer map file.

```
:prob:subs:tot?
15
```

:probeplan:subsites:xoffset?

Short Command

```
:prob:subs:xoff?
```
subsite

Description

This command will return the X offset associated with a given subsite. The X offset is defined as the horizontal distance in microns from the reference point within a die. Offsets are given in the normal Cartesian coordinate system.

Parameters

subsite

Defines the subsite index number. Valid numbers range from 0 to 255.

Example

Retrieves the X offset for the 3rd subsite,

```
:prob:subs:xoff? 3
-4064
```

:probeplan:subsites:yoffset?

Short Command

```
::prob:subs:yoff?
```

subsite

Description

Returns the Y offset associated with a given subsite. The Y offset is defined as the vertical distance in microns from the reference point within a die. Offsets are given in the normal Cartesian coordinate system.

Parameters

subsite

Defines the subsite index number. Valid numbers range from 0 to 255.

Example

Retrieves the Y offset for the 3rd subsite.

:prob:subs:xoff? 3

```
1089
```

:probeplan:test:die

Short Command

```
:prob:test:die
```

X

Y

Mark

Description

This command can be used to mark die for test of the currently loaded wafer map. The X and Y parameters are the column and row in the wafer map user defined coordinates. An optional parameter can be used to deselect a die for test. The testing sequence is recomputed after this command is issued, which might change the testable index of dies that appear after this one.

Parameters

X

Defines the X position or column of the die to mark in user coordinates.

Y

Defines the Y position or row of the die to mark in user coordinates.

Mark

Either ON or OFF depending on whether or not the die is to be marked for test. Use ON to mark the die for test, and OFF to unmark the die for test. If this parameter is not supplied then the default value for mark is.

Example

Sets the die at column 4 and row 5 to be marked for test.

```
:prob:test:die 4 5 ON
```

Deselects the die at column 4 and row 5.

```
:prob:test:die 4 5 OFF
:prob:test:die? 4 5
OFF
```

:probeplan:test:die?

Short Command

```
:prob:test:die?
```

X

Y

Description

This command can be used to query whether or not a has been marked for test in the currently loaded wafer map. The X and Y parameters are the column and row in the wafer map user defined coordinates.

Parameters

X

Defines the X position or column of the die to query in user coordinates.

Y

Defines the Y position or row of the die to query in user coordinates.

Example

Sets the die at column 4 and row 5 to be marked for test.

```
:prob:test:die 4 5 ON
:prob:test:die? 4 5
ON
```

:probeplan:testing:sequence

Short Command

```
:prob:test:seq
```

Direction

Horizontal

Vertical

Description

This command can be used to define the stepping sequence of the currently load wafer map.

Parameters

Direction

Specifies the major axis and bi-direction for the stepping sequence.

HORIZONTAL – single direction along the horizontal.

VERTICAL – single direction along the vertical.

BIDIRHORIZ – Bi-directional along the horizontal axis. Horizontal serpentine motion.

BIDIRVERT – Bi-directional along the horizontal axis. Vertical serpentine motion.

Horizontal

Specifies the direction of the horizontal stepping sequence.

LEFTRIGHT – The horizontal sequence will run from Left to Right.

RIGHTLEFT – The horizontal sequence will run from Right to Left.

Vertical

Specifies the direction of the vertical stepping sequence.

TOPBOTTOM – The vertical sequence will run from Top to Bottom.

BOTTOMTOP – The vertical sequence will run from Bottom to Top.

Example

Sets up a stepping sequence that runs bi-directional with X as the major axis. It runs from Left to Right and Bottom to Top in a horizontal serpentine motion with X as the major axis.

```
:prob:test:seq BIDIRHORIZ LEFTRIGHT BOTTOMTOP
```

:probeplan:testing:sequence?

Short Command

```
:prob:test:seq?
```

Description

This function returns a string describing the current stepping pattern. The string contains three values: Direction, Horizontal, and Vertical.

Possible values for *Direction* are:

HORIZONTAL – single direction along the horizontal.

VERTICAL – single direction along the vertical.

BIDIRHORIZ – Bi-directional along the horizontal axis. Horizontal Serpentine motion.

BIDIRVERT – Bi-directional along the horizontal axis. Vertical Serpentine motion.

Possible values for *Horizontal* are:

LEFTRIGHT – The horizontal sequence will run from Left to Right.

RIGHTLEFT – The horizontal sequence will run from Right to Left.

Possible values for *Vertical* are:

TOPBOTTOM – The vertical sequence will run from Top to Bottom.

BOTTOMTOP – The vertical sequence will run from Bottom to Top.

Parameters

None

Example

Returns that the wafer map stepping sequence is bi-directional with X as the major axis and it runs from Left to Right and Top to Bottom.

```
:prob:test:seq?
```

BIDIRHORIZ LEFTRIGHT TOPBOTTOM

:probeplan:wafer:orientation

Short Command

```
:prob:waf:ori
```

Orientation

Description

This command sets the orientation of the current wafer map. Possible values are 0, 90, 180, 270.
0 is bottom, 90 equals left, 180 equals top, 270 equals right.

Parameters

Orientation

Possible values are 0, 90, 180, 270.

Example

Sets the current orientation of the flat or notch to the left.

```
:prob:waf:ori 90
```

:probeplan:wafer:orientation?

Short Command

`:prob:waf:ori?`
Orientation

Description

This command returns the orientation of the current wafer map. Possible values are 0, 90, 180, 270.

Parameters

None

Example

The wafer map has the flat/notch to the LEFT.

```
:prob:waf:ori?
9
```

:probeplan:wafersize

Short Command

`:prob:waf`
size

Description

This command will set the diameter of the currently loaded wafer map. The size parameter can be in either millimeters or inches, depending on the current units that were set with the `:set:unit` command.

Parameters

size

Defines the diameter of the wafer in the currently loaded wafer map. This parameter can be in either metric (mm.) or English (in.) units depending on the :set:unit command.

Example

Sets the diameter of the current wafer map to 200 mm.

```
:prob:waf 200
```

:probeplan:wafersize?

Short Command

`:prob:waf?`
size

Description

This query command will return the diameter of the currently loaded wafer map. The value that is returned can be either millimeters or inches, depending on the current units that were set with the `:set:unit` command.

Parameters

size

Defines the diameter of the wafer in the currently loaded wafer map. This parameter can be in either metric (mm) or English (in.) units depending on the :set:unit command.

Example

Queries the diameter of the current wafer map.

```
:prob:waf?
200
```

:probeplan:xy:orientation

Short Command

```
:prob:xy:ori
```

XAxis

YAxis

Description

This command will set the orientation of the X and Y axes in the wafer map. This affects the Row, Column readings of the die. RIGHT or LEFT are acceptable parameters for the X Axis, and describe the direction of the positive axis. For the Y Axis UP and DOWN are possible values.

Parameters

XAxis

Acceptable values are RIGHT, LEFT.

YAxis

Acceptable values are UP, DOWN.

Example

Sets the positive X axis RIGHT and Y axis UP.

```
:prob:xy:ori RIGHT UP
```

:probeplan:xy:orientation?

Short Command

```
:prob:xy:ori?
```

Description

This command queries the positive direction of the X and Y-axis of the currently loaded wafer map. Possible return values for the X-axis are RIGHT and LEFT. Return values for the Y-axis are UP and DOWN.

Parameters

None

Example

Returns the values of RIGHT for X and UP for Y.

```
:prob:xy:ori?
RIGHT UP
```

Reverse Orientation Command

Commands
:reverse:orientation

:reverse:orientation

Short Command

`:rev:ori`
devID
[on]
[off]

Description

This command reverses the polarity of the user coordinate system. This reversal only applies to the x- and y-axes.

Parameters

devID
2 for Elite, Summit, S300 and Alessi 6100 emulation. This command can only be used with the chuck channel.

[on] [off]
The On/Off parameter is Off for the standard coordinate space and on for reversed coordinate space. The default setting is Off.

Example

```
:rev:ori 2 on
:rev:ori 2 off
```

Related Commands

:move:absolute
:move:absolute
:move:absolute:chuck
:move:absolute:options
:move:relative
:move:relative:chuck

Set Commands

Commands			
:set:auxiliary	:set:contact:speed	:set:joystick:mode	:set:salign?
:set:busy?	:set:contact:speed?	:set:joystick:mode?	:set:separate
:set:chamber:purge	:set:contact:use:zprofile	:set:light	:set:separate?
:set:chamber:purge?	:set:contact:use:zprofile?	:set:light?	:set:station?
:set:compensation	:set:delay	:set:manual	:set:unit
:set:compensation:factor	:set:delay?	:set:manual?	:set:unit?
:set:compensation:mode	:set:edgesense	:set:microscope:agc	:set:usedeviceid
:set:contact	:set:edgesense?	:set:micr:agc?	:set:usedeviceid?
:set:contact?	:set:edgesense:overdrive	:set:microscope:auto:illuminate	:set:vacuum
:set:contact:active	:set:edgesense:overdrive?	:set:microscope:exposure	:set:vacuum?
:set:contact:active?	:set:edgesense:switch	:set:microscope:exposure?	:set:vacuum:auxiliary
:set:contact:band	:set:edgesense:switch?	:set:microscope:zoom	:set:vacuum:auxiliary?
:set:contact:band?	:set:inker	:set:microscope:zoom?	:set:vacuum:ring
:set:contact:mode	:set:joystick:enabled	:set:preset	:set:vacuum:ring?
:set:contact:mode?	:set:joystick:enabled?	:set:salign	:set:velocity

:set:auxiliary

Short Command

`:set:aux`
[on] [off]

Description

This command turns the power to the probe station auxiliary outlet on or off.

Parameters

[on] [off]
Specifies whether auxiliary power on the station is on or off. The default setting is off.

Example

Turns on auxiliary power on.

```
:set:aux on
```

:set:busy?

Short Command

:set:bus?

Description

This query checks to see if the probe station is currently performing any operations.

Parameters

[true] [false]
Specifies whether auxiliary power on the station is on or off. The default setting is off.

Example

The following shows this command sent to an idle probe station.

```
:set:busy?
FALSE
```

:set:chamber:purge

Short Command

`:set:cham:purg`
Purge State

Description

Controls the amount of MicroChamber air purge. This command is available only on the Elite 300 probe station.

Parameters

Purge State

OFF – Turns off MicroChamber air purge.

MANUAL – The air flow meter on the station is responsible for the amount air flow to the chamber.

QUICK – Sets the purge state to maximum air flow.

AUTO – The software is in charge of controlling the amount of air flow to optimize temperature transitions.

Example

Sets the air chamber purge state to QUICK.

```
:set:chamber:purge QUICK
```

Related Commands

`:set:chamber:purge?`

:set:chamber:purge?

Short Command

`:set:cham:purg?`

Description

Returns the current MicroChamber air purge setting. Possible values are OFF, MANUAL, QUICK, and AUTO. See :set:chamber:purge. This command is only available on the Elite probe station.

Parameters

None

Example

Returns that the system is using QUICK purge.

```
:set:chamber:purge?
QUICK
```

Related Commands

`:set:chamber:purge`

:set:compensation

Short Command

`:set:comp`
[on]
[off]
[thermal]

Description

This command turns compensation mode on or off. When turned on, the currently defined compensation factor is used to correct for thermal expansion or other device-position variables. The correction is applied to `:move:absolute`, `:move:relative`, and `:move:probeplan` commands. It is also applied to device-position queries.

This command functions identically to the `:set:compensation:mode` command.

Generally, you should use the Velox menus to calculate compensation factors. Or, you can numerically set the compensation factors with the `:set:compensation:factor` command.

Parameters

[on]

Specifies whether or not to use the compensation mode. This parameter can have the value on, off or thermal.

[off]

The default value is off.

[thermal]

The thermal mode will use a coefficient of expansion along with the current temperature to dynamically calculate new temperature compensation factors. The coefficient of expansion can be set in the Velox software.

Example

Turns on compensation.

```
:set:comp on
```

:set:compensation:factor

Short Command

`:set:comp:fact`
xFact
yFact

Description

This command numerically sets the compensation factors for movement along the x- and y-axes. The compensation factor is a value by which device moves are multiplied in order to compensate for thermal expansion and other device position variables.

Use the `:set:comp` command to turn on compensation.

If you do not know the necessary compensation factors, you can use Velox menu operations to measure the difference between the nominal and actual position of a test site and calculate the compensation factors.

Parameters

xFact

The value used to calculate the distance the device should move along the x-axis (that is, right or left movement). The nominal distance is multiplied by this value to get the actual distance.

yFact

The value used to calculate the distance the device should move along the y-axis (that is, forward or backward movement). The nominal distance is multiplied by this value to get the actual distance.

Error Condition

An error results if the compensation factor causes device movement parameters to fall outside the limits of device travel.

Example

The following example sets the x- and y-axis compensation factors to increase the move distance by 1 percent along each axis, then turns on compensation mode, moves the device, and queries for the new location. Note that the returned device location is the same as the nominal location, not the actual position with compensation applied. The compensation factor is also applied to the query so that the use of compensation is not reflected in the display of coordinates.

```
:set:comp:fact 1.01 1.01
:set:comp:mode on
:mov:abs 2 +11980 -34 none
:mov:abs? 2
+011980 -000034
```

Related Commands

ALL *:move:absolute* commands
ALL *:move:probeplan:absolute:die* commands
ALL *:move:relative* commands
:set:compensation:mode
:set:compensation

:set:compensation:mode

Short Command

`:set:comp:mode`
[on]
[off]
[thermal]

Description

This command turns compensation mode on or off. When turned on, the currently defined compensation factor is used to correct for thermal expansion or other device-position variables. The correction is applied to `:move:absolute`, `:move:relative`, and `:move:probeplan commands`. It is also applied to device-position queries.

This command functions identically to the `:set:compensation` command.

Generally, you should use the Velox menus to calculate compensation factors. Or, you can numerically set the compensation factors with the `:set:compensation:factor` command.

Parameters

[on]
Specifies whether or not to use the compensation mode. This parameter can have the value on, off or thermal.

[off]
The default value is off.

[thermal]
The thermal mode will use a coefficient of expansion along with the current temperature to dynamically calculate new temperature compensation factors. The coefficient of expansion can be set in the Velox software.

Example

Turns on compensation mode.

```
:set:comp:mode on
```

Related Commands

ALL *:move:absolute* commands
ALL :move:probeplan commands
ALL *:move:relative* commands
:set:compensation:mode
:set:compensation

:set:contact

Short Command

`:set:cont`
devID
contact

Description

This command will set the contact location for the z-stage, and will set the contact location for each positioner used. When used for the z-stage, it defines the location that the DUT comes in contact with the probes. This parameter can be in either microns or mils, depending on the current units that were set with the `:set:unit` command. When used for the positioners, a second parameter is not used and the z-contact position defaults to the current position and the z-axis is reset to zero.

NOTE

When contact zones are enabled, this command will set the contact height for the zone that the stage is currently in. For example, if the stage is positioned so that auxilliary chuck 1 is in the probing area, then the contact height for that zone will be affected.

Parameters

devID

2 for the chuck on Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs. (not applicable to microscope).

contact

For the chuck only, defines the z-stage location that the DUT comes in contact with the probes. This parameter can be in either Metric (microns) or English (Mil) units depending on the :set:unit command.

Example

Turns on compensation.

```
:set:comp on
```

:set:contact?

Short Command

`:set:cont?`
devID

Description

This command will return the contact position of the Z stage. This parameter can be in either microns or mils, depending on the current units that were set with the `:set:unit` command.

NOTE

When contact zones are enabled, this command will return the contact height for the zone that the stage is currently in. For example, if the stage is positioned so that auxilliary chuck 1 is in the probing area, then the contact height for that zone will be returned.

Parameters

devID

2 for the chuck on Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs. (not applicable to microscope).

Example

Shows that the contact location is at 5000 microns.

```
:set:unit?
metric
:set:contact? 2
5000
```

:set:contact:active

Short Command

```
:set:cont:acti
```

Mode

On
Off

Description

This command turns Contact mode on or off.

Prompt is an optional parameter that can be used to enable or disable dialog prompts during Z-Profile setups.

Parameters

Mode

On
Off

Prompt

On – Enables prompts to be displayed when turning on Z Profile
Off – Disables all prompting. This parameter is optional and will default to ON if not provided.

Example

Enables contact mode and disables the prompt dialogs.

```
:set:contact:active ON OFF
```

Related Commands

:set:contact:mode
:set:contact:use:zprofile
:set:contact:use:zprofile?

:set:contact:active?

Short Command

```
:set:cont:acti?
```

Description

This command returns Contact mode status: On or Off.

Parameters

None

Example

Returns that the Contact mode is currently active.

```
:set:contact:active?
ON
```

:set:contact:band

Short Command

`:set:cont:ban`
Band

Description

This command sets the Contact/Search band for Contact mode. Use the `:set:unit` command to specify the units being used.

Parameters

Band

Band value in units: defined by the `:set:unit` command.

Example

Sets the units to metric; sets the contact band to 500 microns; and then checks the command state.

```
:set:unit metric
:set:contact:band 500
:set:contact:band?
500
```

:set:contact:band?

Short Command

`:set:cont:ban?`

Description

This command returns the current Contact/Search band for Contact mode. The command can be used to return the units that the command is using.

Parameters

None

Example

This command returns the Contact/Search band value.

```
:set:contact:band?
500
```

:set:contact:mode

Short Command

`:set:cont:mode`
Standard
Edgesense
Programmable

Parameters

Standard
Sets the mode to Standard.

Edgesense
Turns on the Hardware Edge Sense mode.

Programmable
Sets the Contact mode to Programmable.

Example

Sets the contact mode to programmable and then checks the command state.

```
:set:contact:mode PROGRAMMABLE
:set:contact:mode?
PROGRAMMABLE
```

:set:contact:mode?

Short Command

`:set:cont:mode?`

Description

This command returns a value: Standard; Edgesense; Programmable, depending on the active mode.

Parameters

None

Example

Returns the contact mode that is currently set.

```
:set:contact:mode?
PROGRAMMABLE
```

:set:contact:speed

Short Command

`:set:cont:spee`
Speed

Description

This command sets the search speed used for the Z-axis in Contact mode. Use the `:set:unit` command to specify the units to be used.

Parameters

Speed

Speed value in units defined by the `:set:unit` command.

Example

Sets contact speed to 500 microns per second and then checks the speed.

```
:set:unit metric
:set:contact:speed 500
:set:contact:speed?
500
```

:set:contact:speed?

Short Command

`:set:cont:spee?`

Description

This command returns the current search speed for the Z-axis in Contact mode. The `:set:unit` command may be used to set the returned units.

Parameters

None

Example

Returns the contact speed.

```
:set:contact:speed?
500
```

:set:contact:use:zprofile

Short Command

```
:prof:cont:use:zpr
```

Mode

Description

Enables Z-Profile functionality when programmable contact mode is activated. Parameters must be ON or OFF.

Parameters

Mode

On – Enables Z-Profile in programmable contact mode.

Off – Disables Z-Profile.

Example

Turns on Z-Profile.

```
:set:contact:use:profile ON
```

Related Commands

:set:contact:mode
:set:contact:active
:set:contact:use:zprofile?

:set:contact:use:zprofile?

Short Command

```
:prof:cont:use:zpr?
```

Description

Returns ON if Z-Profile is enabled in programmable contact mode. OFF is returned if Z-Profile is not enabled.

Parameters

None

Example

Returns that Z-Profile has been enabled with programmable contact mode.

```
:set:contact:use:profile?
ON
```

Related Commands

:set:contact:mode
:set:contact:active
:set:contact:use:zprofile

:set:delay

Short Command

`:set:del`
milliseconds

Description

This command specifies a time in milliseconds for the probe station to delay after each operation. During the delay, the probe station does not execute any commands.

Parameters

milliseconds

Specifies the length of time, in milliseconds, which the probe station will delay before executing the next command. You can specify a delay time in the range 0 to 65535 msec (65.5 seconds). By default, the delay is 0.

Example

The first line specifies a delay time of 2 seconds and the second line specifies a delay time of 1 minute.

```
:set:del 2000
:set:del 60000
```

Related Commands

`:set:delay?`
`:system:delay`

:set:delay?

Short Command

`:set:del?`

Description

This query returns the current delay time. The delay is the number of milliseconds the probe station waits after each operation before executing the next command.

Use the `:set:delay` command to set the delay.

Parameters

None

Example

Shows a delay time of 2 seconds:

```
:set:del?
2000
The following example shows a delay time of 1 minute:
:set:del?
60000
```

Related Commands

`:set:delay?`
`:system:delay`

:set:edgesense

Short Command

`:set:edg`
devID
[on] [off]

Description

This command turns edge sense mode on or off. When on, the chuck up position is the point at which the DUT makes contact with an edge sense probe plus any specified overdrive distance. Using edge sense makes it easier to raise the chuck to the optimal probing height.

To set the parameters required to operate in edge sense mode, use the `:set:edgesense:overdrive` and `:set:edgesense:switch` commands.

Parameters

devID
2 for Elite, Summit, S300 and Alessi 6100 emulation. Applicable to chuck only.

[on] [off]
Use On to activate edge sense mode and Off to turn off edge sensing.

Example

The following example sets the non-contacted position of the edge sense probe, the amount of overdrive, and turns edge sense mode on. In this example, the amount of overdrive is set to 75.

```
:set:edg:swit 2 nopen
:set:edg:over 2 75
:set:edg 2 on
```

Related Commands

`:set:edgesense:overdrive`
`:set:edgesense:switch`
`:move:contact`

:set:edgesense?

Short Command

`:set:edg?`
devID
[on] [off]

Description

Returns whether or not contact mode is enabled.

Parameters

devID
2 for Elite, Summit, S300 and Alessi 6100 emulation. Applicable to chuck only.

Example

The following example returns the status of edge sense: on or off.

```
:set:edg? 2
ON
```

Related Commands

`:set:edgesense:overdrive`
`:set:edgesense:switch`
`:move:contact`

:set:edgesense:overdrive

Short Command

`:set:edg:over`
devID
microns

Description

This command sets the distance the chuck rises after contact is made with the edge sense probe. This command affects chuck movement only when the probe station is operating in edge sense mode.

Parameters

devID
2 for Elite, Summit, S300 and Alessi 6100 emulation. Applicable to chuck only.

microns
Specifies the distance the chuck will rise after contacting the edge sense probe.

Example

The following example sets the non-contacted position of the edge sense probe, the amount of overdrive, and turns edge sense mode on.

```
:set:edg:swit 2 nopen
:set:edg:over 2 2
:set:edg 2 on
```

Related Commands

`:set:edgesense:overdrive`
`:set:edgesense:switch`
`:move:contact`

:set:edgesense:overdrive?

Short Command

`:set:edg:over?`

Description

This command returns the current overdrive amount that is set in Contact mode. The `:set:unit` command may be used to set the returned units.

Parameters

None

Example

Returns that the overdrive amount is 10 microns.

```
:set:edgesense:overdrive?
10
```

Related Commands

`:set:edgesense:overdrive`
`:set:edgesense:switch`
`:move:contact`

:set:edgesense:switch

Short Command

`:set:edg:swit`
devID
[nopen]
[nclosed]

Description

This command identifies the non-contacted position of the switch in the edge sense probe. For example, if the switch opens when the DUT touches the probe tip, then the non-contacted position is closed.

Parameters

devID
2 for Elite, Summit, S300 and Alessi 6100 emulation. Applicable to chuck only.

[nopen]
Use nopen if the switch is normally open.

[nclosed]
Use nclosed if the switch is normally closed.

Example

The following example sets the noncontacted position of the edge sense probe, the amount of overdrive, and turns edge sense mode on.

```
:set:edg:swit 2 nopen
:set:edg:over 2 2
:set:edg 2 on
```

Related Commands

`:set:edgesense:overdrive?`
`:set:edgesense:switch?`
`:set:edgesense`
`:move:contact`
`:set:edgesense:overdrive`

:set:edgesense:switch?

Short Command

`:set:edg:swit?`

Description

This command returns the current switch type that is defined for use with edge sense. Possible values are: NOSWITCH, NOPEN, or NCLOSED.

Parameters

None

Example

Returns that the switch type is normally closed.

```
:set:edgesense:switch?
NCLOS
```

Related Commands

:set:edgesense:overdrive?
:set:edgesense:switch?
:set:edgesense
:set:edgesense:overdrive

:set:inker

Short Command

`:set:ink`
Length

Description

This command can be used to fire the inker. The [*Length*] parameter is the time in milliseconds that the inker is activated.

Parameters

Length
Amount of time inker is kept activated in milliseconds. Valid is 0 to 2001.

Example

Fires the inker for 100 milliseconds.

```
:set:ink 100
```

:set:joystick:enabled

Short Command

`:set:joys:enab`
enabled
[on]
[off]

Description

This command can be used to enable or disable the joystick. When the joystick is disabled it cannot control stage movements.

Parameters

enable
Possible values are either On or Off. On will enable the joystick and Off will disable the joystick.

Example

Disables the joystick.

```
:set:joystick:enabled OFF
```

:set:joystick:enabled?

Short Command

`:set:joys:enab?`

Description

This command will query whether or not the joystick is enabled. Possible return values are on or off. If on is returned the joystick is enabled, and if off is returned the joystick is disabled.

Parameters

None

Example

Shows that the joystick has been disabled.

```
:set:joystick:enabled? OFF
```

:set:joystick:mode

Short Command

```
:set:joys:mode
```
mode

Description

Changes the motion control mode that is used by the joystick. This is the same as selecting a different mode in the combo box drop down on the motion control window. See Velox Help, ControlCenter topics.

Parameters

mode

All motion modes are valid: SCAN, SCANAUTOZ, SCANZ, INDEX, DIE, SUBINDEX, JOG, THETA.

Example

Sets the joystick mode to INDEX.

```
:set:joystick:mode INDEX
```

Related Commands

:set:joystick:mode?

:set:joystick:mode?

Short Command

```
:set:joys:mode?
```

Description

Returns the currently defined motion mode. Possible values are SCAN, SCANAUTOZ, SCANZ, INDEX, DIE, SUBINDEX, JOG, THETA. See Velox Help, ControlCenter topics.

Parameters

None

Example

Returns that the joystick is currently in INDEX mode.

```
:set:joystick:mode?
INDEX
```

Related Commands

:set:joystick:mode

:set:light

Short Command

```
:set:ligh
```
[on]
[off]

Description

This command turns the microscope light on or off.

Parameters

[on] [off]

Specifies whether the microscope light for the probe station optics is on or off. By default, the light is off.

Example

Shows how to turn the microscope light on.

```
:set:ligh on
```

:set:light?

Short Command

```
:set:ligh?
```

Description

This command returns the current state of the microscope light.

Parameters

None

Example

Turns the light on and off querying the state after each command.

```
:set:light ON
:set:light?
ON
:set:light OFF
:set:light?
OFF
```

Related Commands

:set:light

:set:manual

Short Command

```
:set:man
```

devID

on]

[off]

Description

This command toggles from semiautomatic to manual mode. The station must be in manual mode before using the x- and y-axis knobs on the probe station to move the device. The status window continues to track and display the current device location while in manual mode. Manual Mode is only available on the chuck channel of motion.

When you exit manual mode, Velox lowers the chuck, moves the motors momentarily, and raises the chuck again.

Parameters

devID

2 for Elite, Summit 12000-Series and S300.

on] [off]

Specifies whether manual mode is on or off. The default is off.

Example

Shows how to turn on manual mode.

```
:set:man 2 on
```

Related Commands

:set:manual?

:set:manual?

Short Command

```
:set:man?
```

devID

on]

[off]

Description

This query returns ON if the probe station is operating in manual mode and OFF if operating in semiautomatic mode.

Parameters

devID

2 for Elite, Summit 12000-Series and S300.

Example

Shows how to turn on manual mode and query for the status.

```
:set:man 2 on
:set:man? 2
ON
```

Related Commands

:set:manual

:set:microscope:agc

Short Command

```
:set:micr:agc
```

Description

This commands enables or disables the use of Automatic Gain Control (AGC) for the microscope.

Parameters

Mode

ON enables AGC

OFF disables AGC

Example

This command turns on Automatic Gain Control (AGC):

```
:set:micr:agc on
```

Related Commands

:set:micr:agc?

:set:micr:agc?

Short Command

`:set:micr:agc?`

Description

This commands queries the current state of Automatic Gain Control for the microscope.

Parameters

None

Example

This command checks AGC:

`:set:micr:agc?`

Related Commands

`:set:microscope:agc`

:set:microscope:auto:illuminate

Short Command

`:set:micr:auto:ill`
Contrast/Brightness
On
Off

Description

Supported only on an eVue system. Runs an auto-illuminate routine at the current AF bounding box. Contrast/Brightness is an optional parameter. If it is not provided, then Brightness, Exposure, and Contrast are all adjusted.

Parameters

Contrast/Brightness
ON or OFF. Defines whether or not Contrast and Brightness are adjusted along with the Exposure.

- If this parameter is OFF, then only Exposure is adjusted.
- If this parameter is ON, then all are adjusted.
- If parameter is not provided, then ON is used.

Example

Runs an auto illumination at the current site.

```
:set:microscope:auto:illuminate
COMPLETE
```

Related Commands

`:set:microscope:exposure`
`:set:microscope:exposure?`

:set:microscope:exposure

Short Command

`:set:micr:exp`
Exposure

Description

Supported only on an eVue system. This command sets the amount of exposure that is used by the CCDs of an eVue microscope. Exposure rate of the CCDs has an effect on the frame rate. Faster exposure rates correspond to less light, but mean faster frames per second.

Parameters

Exposure

Set the level of exposure used for the eVue microscope in milliseconds. Valid range is 0.5 to 100 ms.

Example

Sets the exposure rate to 30 milliseconds.

```
:set:microscope:exposure 30
complete
```

Related Commands

:set:microscope:exposure?

:set:microscope:exposure?

Short Command

```
:set:micr:exp?
```

Description

Supported only on an eVue system. This command sets the amount of exposure that is used by the CCDs of an eVue microscope. Exposure rate of the CCDs has an effect on the frame rate. Faster exposure rates correspond to less light, but mean faster frames per second.

Parameters

None

Example

Returns that the exposure rate is set to 30 milliseconds.

```
:set:microscope:exposure?
30
```

Related Commands

:set:microscope:exposure

:set:microscope:zoom

Short Command

```
:set:micr:zoom
```
Zoom

Description

Valid only with the eVue microscope or an A-Zoom microscope that is connected to Velox. On A-Zoom systems, this command sets the zoom value used by the microscope. On eVue systems, this command sets the proper CCD zoom level to the appropriate optical path.

Parameters

Zoom

Sets the current zoom level of the microscope.

A-Zoom: valid ranges are 0 to 99.

eVue 10x: valid ranges are 0.5 to 5.0.

eVue 40x: valid ranges are 0.5 to 20.0.

Example

Sets the zoom value to 35 and checks the command state.

```
:set:microscope:zoom 35
:set:microscope:zoom?
35
```

Related Commands

`:set:microscope:zoom?`

:set:microscope:zoom?

Short Command

`:set:micr:zoom?`

Description

Valid only with the eVue microscope or an A-Zoom microscope that is connected to Velox. On A-Zoom systems, this command returns an integer value from 0 to 99, which is the zoom value set for the microscope. On eVue systems, this command, returns values ranging from 0.5 to 5.0 for a 10x system and 0.5 to 20.0 for a 40x system. The CCD parameter can be used to query a specific optical path in Multi-Camera mode.

Parameters

CCD

The CCD parameter is valid only on eVue systems and is an optional parameter. Valid CCD parameters are 1, 2, or 3. If not provided, then CCD 1 is used.

Example

Returns a zoom value on CCD2 of an eVue system.

```
:set:microscope:zoom? 2
2.5
```

Related Commands

`:set:microscope:zoom`

:set:preset

Short Command

`:set:pres`
devID
x
y

Description

This command presets the coordinate system coordinates for `:move:absolute` device movements by assigning the specified user coordinates to the current device location. You can assign any valid user coordinates to the current device position.

NOTE

*The `:set:preset` command and the user values of status window in the UI are tied together. For example, pressing Set Zero in the status window will make a call to **`:set:preset`**, and :set:preset will change the user values in the status window.*

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

x

Identifies the x coordinate that will be assigned to the current device location. The value of the parameter is interpreted as either mils or microns. Use the `:set:unit` command to switch between mils and microns.

By default, the user coordinate system has the same origin as the probe-station machine coordinate system.

y

Identifies the y coordinate that will be assigned to the current device location. The valid range of values for this command is the same as for the x parameter.

Example

Sets the user 0,0 location. This example assumes that you have already positioned the device at the location to which you want to assign the user 0,0. You can use `:move:abs?` to return the current device location, as shown here:

```
:mov:abs? 2
+001158 -003341 000400
:set:pres 2 0 0
:mov:abs? 2
+000000 +000000 000400
```

The following example shows how to assign coordinates x=150, y=281 to the current device location:

```
:set:pres 2 150 281
:mov:abs? 2
+000150 +000281 000400
```

Related Commands

`:move:absolute`
`:move:absolute?`
`:move:absolute:chuck`
`:move:absolute:options`

:set:salign

Short Command

`:set:sal`
devID
[on] [off]
x offset
y offset

Description

This command turns on and off software alignment mode. When software alignment is on, Velox adds offsets to device movements to compensate for rotational misalignment between the DUT and the probe station axes. This is an alternative to aligning the hardware by adjusting theta.

The x offset and y offset parameters are optional. If they are provided then Velox will use the angle created by x offset and y offset as the alignment angle. If these parameters are not given then the angle that was set up in Velox is used.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. Applicable to chuck only.

[on] [off]

Specifies whether software angle compensation is on or off. Use the character strings on and off for this parameter.

x offset

Used along with the y offset parameter to define the angle of software correction. This parameter is given in microns. This parameter is optional. If the x offset parameter is given then the must also be given.

y offset

Used along with the x offset parameter to define the angle of software correction.

Example

The following command turns on software alignment mode:

```
:set:sal 2 on
```

Related Commands

:set:salign?

:set:salign?

Short Command

```
:set:sal?
```

devID

Description

This query returns on if the probe station is operating in software alignment mode and off if it is not.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. Applicable to chuck only.

Example

The following commands turns on software alignment mode, then queries for the status:

```
:set:sal 2 on
:set:sal? 2
ON
```

Related Commands

:set:salign

:set:separate

Short Command

```
:set:sep
```

devID

separate

Description

This command will set the separate distance away from contact used in auto Z stepping. It defines the distance away from contact that the DUT needs to be in order to clear the probes. Since separate is a distance, it must be a positive number. This parameter can be in either microns or mils, depending on the current units that were set with the `:set:unit` command.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs (all but microscope).

separate

Defines the Z distance to travel away from contact so the DUT is clear of the probes. This parameter must be greater than 0 because if defines a distance.

Example

Sets separate distance at 500 microns.

```
:set:unit metric
:set:sep 2 500
```

:set:separate?

Short Command

`:set:sep?`
devID

Description

This command will return the separate distance of the Z stage. This parameter can be in either microns or mils, depending on the current units that were set with the `:set:unit` command.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation.See SCPI Device IDs (all but microscope).

Example

Shows that the separate distance is 500 microns.

```
:set:unit?
metric
:set:separate? 2
500
```

:set:station?

Short Command

`:set:stat?`

Description

This command will return 2, which is the number for Summit 12000 B-Series probe stations.

Parameters

None

Example

The following query line requests the identification number from a device in the system. The returned character (2) indicates that the device is a Summit 12000 B-Series probe station.

```
:set:stat?
2
```

:set:unit

Short Command

`:set:unit`
[metric]
[English]
[metric_float]

Description

This command specifies the units of measurement that will be used for all input and output fields. For example, if you set the units of measurement to metric units (the default setting), x- and y-axis coordinates are in microns, and the wafer diameter is in mm. If you set the units of measurement to English units, the x- and y-axis coordinate values are in mils and the wafer diameter is in inches.

Parameters

[metric]

Specifies that metric units of measurement will be used for distances and other values. By default, the units are metric. Parameters can be abbreviated to metr.

[English]

Specifies that English units of measurement will be used for distances and other values. By default, the units are metric. Parameters can be abbreviated to engl.

[metric_float]

Specifies that metric units of measurement will be used for distances and other values. In addition the METRIC_FLOAT unit type will return microns with 2-digit precision from commands like :move:absolute?

Example

Sets English as the units of measurement.

```
:set:unit English
```

Related Commands

:set:unit?
:move:absolute:options
:move:absolute
:move:absolute
:set:preset

:set:unit?

Short Command

```
:set:unit?
```

Description

This query command returns the unit of measurement which is being used by distances and coordinates previously through the `:SET:UNIT` command. It either returns “METRIC”, “ENGLISH” or METRIC_FLOAT as a response.

Parameters

None

Example

Shows how to determine the current unit of measurement.

```
:set:unit?
METRIC
```

Related Commands

:set:unit

:set:usedeviceid

Short Command

`:set:used`
active
[on]
[off]

Description

This command is obsolete. It is accepted and will not generate an error. All the commands controlled by this command will accept the deviceID parameter, but ignore it. For all new implementations, we recommend not using deviceID unless it is documented as required.

This command used to control the use of the device ID parameter (*devID*) for many SCPI commands. When the value was set on, the device ID was used in the affected commands. When the value was off, the ID was not used in the affected commands.

Parameters

active
Must be either on or off, but the value is ignored.

:set:usedeviceid?

Short Command

`:set:used?`

Description

This command is now obsolete. It will always return OFF. All commands previously affected by this command will always accept, but ignore Device ID.

Parameters

None

:set:vacuum

Short Command

`:set:vac`
[on]
[off]

Description

This command turns the vacuum on or off in the center of the chuck. This command does not affect the corner (or auxiliary-device) vacuum. Nor does it affect the size of the DUT hold-down area.

The corner vacuum controls and the size control for the center vacuum area are manual. Any vacuum ports not covered by the DUT should be turned off for maximum efficiency.

Parameters

[on]
Specifies that vacuum to the chuck top is on or off (the default). When set to On, this command turns on the vacuum to the chuck top.

[off]
When set to Off, this command turns off the chuck top vacuum.

Example

Shows how to turn on vacuum to the chuck top:

```
:set:vac on
```

Related Commands

:set:unit?

:set:vacuum?

Short Command

```
:set:vac?
```

Description

Returns ON or OFF depending on the state of the vacuum output line (not a sense line). Use the command :system:vacuum:sense to detect if the system actually has vacuum or not.

Parameters

None

Example

Returns that the vacuum control line is turned on.

```
:set:vacuum?
ON
```

Related Commands

:set:vacuum
:set:vacuum:auxiliary
:set:vacuum:auxiliary?
:set:vacuum:ring
:set:vacuum:ring?
:system:vacuum:sense?

:set:vacuum:auxiliary

Short Command

```
:set:vac:aux
```

Description

Used to turn vacuum on or off to the auxiliary chucks. The auxiliary chuck vacuum operates independently from the master chuck vacuum. Only valid on the Elite probe station.

Parameters

Aux. Chuck

Possible values are 1 or 2.

Enable

Possible values are ON or OFF.

Example

Shows how to turn vacuum on to the first auxiliary chuck.

```
:set:vacuum:auxiliary 1 ON
```

Related Commands

:set:vacuum

:set:vacuum:auxiliary?

Short Command

`:set:vac:aux?`

Description

Returns “ON” or “OFF” depending on the vacuum control line for the requested auxiliary chuck. Only valid on the Elite probe station.

Parameters

Aux. Chuck

Defines which Auxiliary chuck have vacuum turned on or off. Possible values are 1 or 2.

Example

Returns “ON” if the vacuum control line is active.

```
:set:vacuum:auxiliary? 1
ON
```

Related Commands

:set:vacuum
:set:vacuum?
:set:vacuum:auxiliary
:set:vacuum:ring
:set:vacuum:ring?

:set:vacuum:ring

Short Command

`:set:vac:rin`

Description

Used to enable vacuum to the rings of the chuck. The vacuum rings are dependant on master vacuum. If master vacuum is off and a vacuum ring is enabled, then it does not turn on vacuum to the ring. If master vacuum is on, then it does turn vacuum on to the specified ring. Actual vacuum is controlled with :set:vacuum command. Only valid on the Elite probe station.

Parameters

Ring Size

Possible values are 75MM, 150MM, 200MM, 300MM.

Enable

Possible values are ON or OFF.

Example

The first example enables the 75MM vacuum ring. If the master vacuum is ON, then vacuum to this ring is turned on. The second example disables the 300MM vacuum ring.

```
:set:vacuum:ring 75MM ON
:set:vacuum:ring 300MM OFF
```

Related Commands

:set:vacuum
:set:vacuum?
:set:vacuum:auxiliary
:set:vacuum:auxiliary?
:set:vacuum:ring?

:set:vacuum:ring?

Short Command

`:set:vac:rin?`

Description

Returns “ON” or “OFF” depending on the enable state of the specified vacuum ring. Only valid on the Elite probe station.

Parameters

Ring Size

Possible values are 75MM, 150MM, 200MM, 300MM.

Example

Returns “ON” if the vacuum to that ring is enabled.

```
:set:vacuum:ring? 75MM
ON
```

Related Commands

:set:vacuum
:set:vacuum?
:set:vacuum:auxiliary
:set:vacuum:auxiliary?
:set:vacuum:ring

:set:velocity

Short Command

`:set:vel`
devID
[xyz]
[ufast]
[vfast]
[fast]
[medium]
[slow]
[vslow]
[uslow

Description

This command specifies the velocity setting for chuck and device movements along all axes on semiautomatic probe stations. This command affects the velocity used for the *:move:absolute*, *:move:relative*, and die stepping commands. This command does not apply to moves made in scan mode.

Parameters

devID

2 for Elite, Summit, S300 and Alessi 6100 emulation. See SCPI Device IDs.

[xyz]

Sets the velocity for horizontal device movements (along the x- and y-axes) or for vertical chuck movements (along the z-axis).

[ufast]

Specifies the velocity as **ufast, vfast, fast, medium, slow, vslow**, or **uslow**. The default value for this parameter is ultra fast (ufast).

Example

Moves the device at slow velocity and raises the chuck at medium velocity.

```
:set:vel 2 xy slow
:set:vel 2 z medium
:mov:abs 2 7000 7000 750
```

Related Commands

:move:absolute
:move:absolute:chuck
:move:relative

System Commands

Commands		
:system:beep	:system:error?	:system:operation:mode?
:system:configuration?	:system:identification?	:system:platen?
:system:delay	:system:limits:travel?	:system:vacuum:sense?
:system:display	:system:operation:mode	:system:version?

:system:beep

Short Command

```
:syst:beep
```

Description

This command causes the system to emit a sound.

Parameters

None

Example

The following causes the system to emit a “beep” sound:

```
:syst:beep
```

:system:configuration?

Short Command

```
:syst:conf?
```

Limits

Channels

Description

Returns a string response containing the system configuration value of the parameter requested. Responses will vary depending on the parameter string.

Parameters

Limits

Returns the limit of stage travel as a single number. Assumes a square stage with the center of travel being 0,0.

Channels

Returns a string containing the currently available channels. The string contains a space between each channel. Ex: “2 3 7” would signify that the Chuck, Scope, and Positioner are available. The IDs returned are the same as the device IDs for the move absolute command: 2 – Chuck, 3 – Scope, 7 – Positioner One, 8 – Positioner Two, up until 12 for Positioner six.

Example

Queries the system for the channels that are available. The return string shows that the Chuck and Scope are the only two channels available.

```
:system:configuration? channels
2 3
```

:system:delay

Short Command

`:syst:del`
msec

Description

This command makes the probe station wait the specified length of time before executing the next instruction. During this delay period, the probe station does not accept commands from application programs or remote hosts.

This is a one-time delay. If you want to specify a delay for the probe station to wait after the execution of every instruction, use the `:set:delay` command.

Parameters

msec

Specifies the length of time, in milliseconds, which the probe station will delay. The delay parameter can be a value in the range 0 to 65535 (65.5 seconds).

Error Condition

For remote communications applications, avoid setting a delay that is longer than your timeout values. Consider using serial polls or SRQ handshaking instead.

Example

The delay features are useful with the `:move:scan` commands. In this example, you want to scan at slow speeds for 2 sec, then stop device movement.

```
:mov:scan:xax 2 right slow
:syst:del 2000
:mov:stop:xax 2
```

Related Commands

`:set:delay`

:system:display

Short Command

`:syst:disp`
string

Description

This command displays a one-line message on the probe station screen. The message remains until you press **Enter**, or click OK on the message box, to confirm the message.

Use this command to insert prompts, warnings, and other messages into a test program.

Parameters

string

The ASCII string representing the message you want to display on the probe station screen. The character string can be up to 50 characters long and must be enclosed in double quotes, for example, “text here is correct.”

The double quotes that delineate the string are not displayed on the probe station screen as part of the string, and you cannot include quotes in the string. Characters following the second double quote are interpreted as a part of your application program or as a part of another command. If you want to include quotes in your message, use single quotes, for example 'quote here' is correct.

Error Condition

If you have a remote control application using timeout values and the time it takes you to respond to the message exceeds the time limit, you will time out the probe station. Consider using serial polls or SQR handshaking instead..

Example

Shows a message that tells the user to perform a particular test. In this example, the second `:move:relative` command executes as soon as you confirm the message Measure S11 by pressing a key on the probe station.

```
:mov:rel 2 14 0 400
:syst:disp "Measure S11. Press enter key."
:mov:rel 2 14 0 400
```

:system:error?

Short Command

`:syst:err?`

Description

This query returns the last error recorded by the probe station. The returned error is an ASCII string. If no error has occurred, this query returns an empty string.

This feature is especially useful when troubleshooting your system. For example, if your application program is not working as expected, you might have created an error situation that your program does not recognize. In this case, you can use the return-error feature to see what error occurred. This can help you trace the error to a particular line in your test program. The error string can be cleared by using an optional parameter. If the error has been cleared and no new error occurs then "COMPLETE" is returned, indicating that no new errors have ocurred since calling this command.

Parameters

Clear Error (optional)

Possible values are ON or OFF. If the parameter is not present then the error is not cleared. A value of ON will clear the error after it is returned.

Example

Uses the `:system:error?` command to see the last error message that occurred.

```
:syst:err?
@The number entered is greater than the upper limit of 6024
```

In the previous case, the last error recorded indicates that you entered (or sent) coordinates for a device movement. However, the coordinates were outside the probe station limit switches.

By using "ON" it returns the current error and then clears it.

```
:system:error? ON
```

@The number entered is greater than the upper limit of 6024

So the next time the command is called, "COMPLETE" is returned.

```
:system:error?
COMPLETE
```

:system:identification?

Short Command

`:syst:iden?`

Description

This query returns the probe station identification string. This is defined in the Station component of the Hardware Configurator.

Parameters

None

Example

Requests the identification string from the probe station. The string returned by your the probe station will differ depending on the probe-station serial number and version.

```
:syst:iden?
Summit 12000
```

:system:limits:travel?

Short Command

`:syst:lim:trav?`
devID

Description

This command will return the limits of travel for the specified channel. Currently only the stage is implemented. Values returned for the X and Y axis are in microns, and the theta axis is in milli-degrees. On systems without motorized theta, 0 will be returned for theta limits. On Summit 12000-Series systems the Z axis limits are in motor counts. The return string will be formatted as follows: +X -X +Y -Y +Z -Z +Theta -Theta.

Parameters

devID

Specifies which channel of motion which to query.

Example

Queries the stage for limits on a Summit 12000-Series system. This system does not have a motorized theta so 0 0 is returned. X and Y each have a limit of +/-103000, the Z axis is 5500 at the top of travel to 0 at the bottom.

```
:syst:lim:trav? 2
+103000 -103000 +103000 -103000 5500 0 0 0
```

:system:operation:mode

Short Command

`:syst:oper:mode`
Mode

Description

Sets Velox software into REMOTE or LOCAL mode. When in REMOTE mode, all windows are hidden except the video window and point & shoot is disabled. Joystick still functions in REMOTE mode. LOCAL mode restores normal operations to the UI and point & shoot is re-enabled.

Parameters

Mode

Possible values are REMOTE or LOCAL.

Example
Sets the system into LOCAL mode. `:system:operation:mode LOCAL`
Related Commands
`:system:operation:mode?`

:system:operation:mode?

Short Command
`:syst:oper:mode?`
Description
Returns REMOTE or LOCAL depending on what mode is currently active.
Parameters
None
Example

Returns that the system is in LOCAL mode.
`:system:operation:mode?`
`LOCAL`

Related Commands
`:system:operation:mode`

:system:platen?

Short Command
`:syst:plat?`
Description
This command returns the status of the platen arm. Possible return values are “UP”, “DOWN” or “PARTIAL”. If “UP” is returned then the platen arm is fully raised. If “DOWN” is returned then the platen is fully lowered. If “PARTIAL” is returned then the platen arm is in-between the up and down positions.
Parameters
None
Example

Shows that the platen arm is fully raised.
`:system:platen?`
`UP`

:system:vacuum:sense?

Short Command
`:syst:vac:sens?`
Description
Returns ON if the system senses vacuum, OFF if it does not sense vacuum. This command can not detect the presence of a wafer, it can only return whether or not the system senses vacuum.
Parameters

None

Example

Returns that vacuum is sensed on the system.

```
:system:vacuum:sense?
ON
```

Related Commands

:set:vacuum?
:set:vacuum

:system:version?

Short Command

```
:syst:vers?
```

Description

This query returns the probe station software release version.

Parameters

None

Example

Shows a sample response when you send the version command to the probe station. In this example, the probe station version number is 4.0.

```
:syst:vers?
4.0
```

Related Commands

:system:identification?

Thermal Chuck Commands

Commands		
:thermalchuck:activate	:thermalchuck:status?	:thermalchuck:temperature:target?
:thermalchuck:activate?	:thermalchuck:temperature:current?	:thermalchuck:temperature:window
:thermalchuck:deactivate	:thermalchuck:temperature:settemp	:thermalchuck:temperature:window?

:thermalchuck:activate

Short Command

```
:ther:acti
```

Description

This command activates the attached thermal controller unit using the current settings.

NOTE

The thermal chuck must first b deactivated before you can set a temperature. After the temperature is set it will not take effect until the system is activated.

Parameters

None

Example

Sets the thermal chuck to become active.

```
:thermalchuck:deactivate
:thermalchuck:temperature:settemp 60.0
:thermalchuck:activate
```

:thermalchuck:activate?

Short Command

```
:ther:acti?
```

Description

This command determines whether the attached thermal controller is in the active or inactive state.

Parameters

None

Example

Checks the thermal chuck for active/inactive status.

:ther:temp:acti?

:thermalchuck:deactivate

Short Command

```
:ther:deac
```

Description

This command deactivates the thermal and sets it to idle.

NOTE

The thermal chuck must first be deactivated before you can set a temperature. After the temperature is set, it will not take effect until the system is activated.

Parameters

None

Example

Sets the thermal chuck to idle.

```
:thermalchuck:deactivate
:thermalchuck:temperature:settemp 60.0
:thermalchuck:activate
```

:thermalchuck:status?

Short Command

```
:ther:stat?
```

Description

This command returns the status of the thermal chuck. Possible return values are: AT TEMP, RUNNING, IDLE, or ERROR, depending on the specific thermal controller used.

Parameters

None

Example

Returns the status of the thermal chuck.

```
:ther:sta?
```

AT TEMP

:thermalchuck:temperature:current?

Short Command

```
:ther:temp:curr?
```

Description

This command returns the current temperature of the thermal chuck in degrees Celsius.

Parameters

None

Example

Returns the thermal chuck temperature.

```
:ther:temp:curr?
```

-34.5

:thermalchuck:temperature:settemp

Short Command

```
:ther:temp:sett
```

value

Description

This command can be used to set the thermal chuck to a temperature. The value is a floating point number in degrees Celsius.

NOTE

The thermal chuck must first be deactivated before you can set a temperature. After the temperature is set, it will not take effect until the system is activated.

Parameters

value

The desired temperature for the thermal chuck.

Example

```
:thermalchuck:deactivate
:thermalchuck:temperature:settemp 60.0
:thermalchuck:activate
```

:thermalchuck:temperature:target?

Short Command

```
:ther:temp:targ?
```

Description

This command returns the current target temperature last set with the `:ther:temp:sett` command. If no “settemp” command has been issued, this value shall be 0.0 degrees Celsius.

Parameters

None

Example

Returns the current target temperature of the thermal chuck.

```
:ther:temp:targ?
-25.4
```

:thermalchuck:temperature:window

Short Command

`:ther:temp:wind`
window

Description

This command sets the window range of the thermal chuck when the unit is in the inactive state. The `:thermalchuck:temperature:window` is not supported on Titan thermal chucks.

Parameters

window
Entire range of window (+0.1 <= WINDOW <= 9.9)

Example

The following command sets the window range to 5.0 degrees Celsius.
:ther:temp:wind 5.0

:thermalchuck:temperature:window?

Short Command

`:ther:temp:wind?`

Description

This command returns the current window range. If the `:ther:temp:wind` command has not been issued, this value shall be 0.2 degrees Celsius. This command is not supported on Titan thermal controllers.

Parameters

window
Entire range of window (+0.1 <= WINDOW <= 9.9)

Example

Returns the current window range of the thermal chuck.

```
:ther:temp:wind?
2.5
```

Vision Commands

Commands		
:vision:search:target	:vision:set:match:score	:vision:train:target

:vision:search:target

Short Command

`:vis:sear:targ`

Description

Performs a pattern req. search using Spectrum Vision. Vision must be purchased and installed on the system before this command will work. In order to use this command the target must first be trained using the Auto XY Correction window. After a search is preformed the return values are the X, Y shift of the target and a match score. If the match score is 0, the target was not found in the field of view. The X and Y values are oriented such that a call to ":move:rel X Y" will move the target directly into the target train box of the live video window. See Auto XY Correction in the Velox Help for more information.

Parameters

None

Example

The following example performs a pattern req. search and returns that the target has moved 30 microns in X and -20 microns in Y with a match score of 830.

```
:vis:sear:targ
30 -20 830
```

Related Commands

:move:relative

:vision:set:match:score

Short Command

`:vis:set:matc:scor`
Match Score

Description

This command can be used to set the threshold score used during vision search operations. If the current search target yields a score that is below this threshold, the vision system returns that no matches could be found. In some cases, it can be effective to lower the match score to yield passing targets. The possible range for match score is 0 to 1000. When the match score is set at 1000, only targets that are a perfect match will succeed. The default setting is 700.

Parameters

Match Score

Defines the threshold score that is used as the criteria for deciding whether or not a target is a good match.

Example

Sets the match score threshold to 700.

```
:vision:set:match:score 700
COMPLETE
```

Related Commands

:vision:train:target
:vision:search:target

:vision:train:target

Short Command

`:vis:tra:targ`
Vision ID

Description

This remote command trains a target using the region defined by the vision box in the video window. Once a target has been trained, you can search for a match using the `:vision:search:target` command. The location of the training region is recorded for each target, which ensures that the offset of multiple trained targets can be managed.

Parameters

Vision ID

Used to identify which target to train. The current location of the vision train box in the video window defines the location where the image is trained. Possible values range from 1 to 50. If the parameter is not provided, then the Auto XY Correction target is trained.

Example

Trains the vision target ID to 12.

```
:vis:trai:target 12
COMPLETE
```

Related Commands

`:vision:train:target`

Appendix

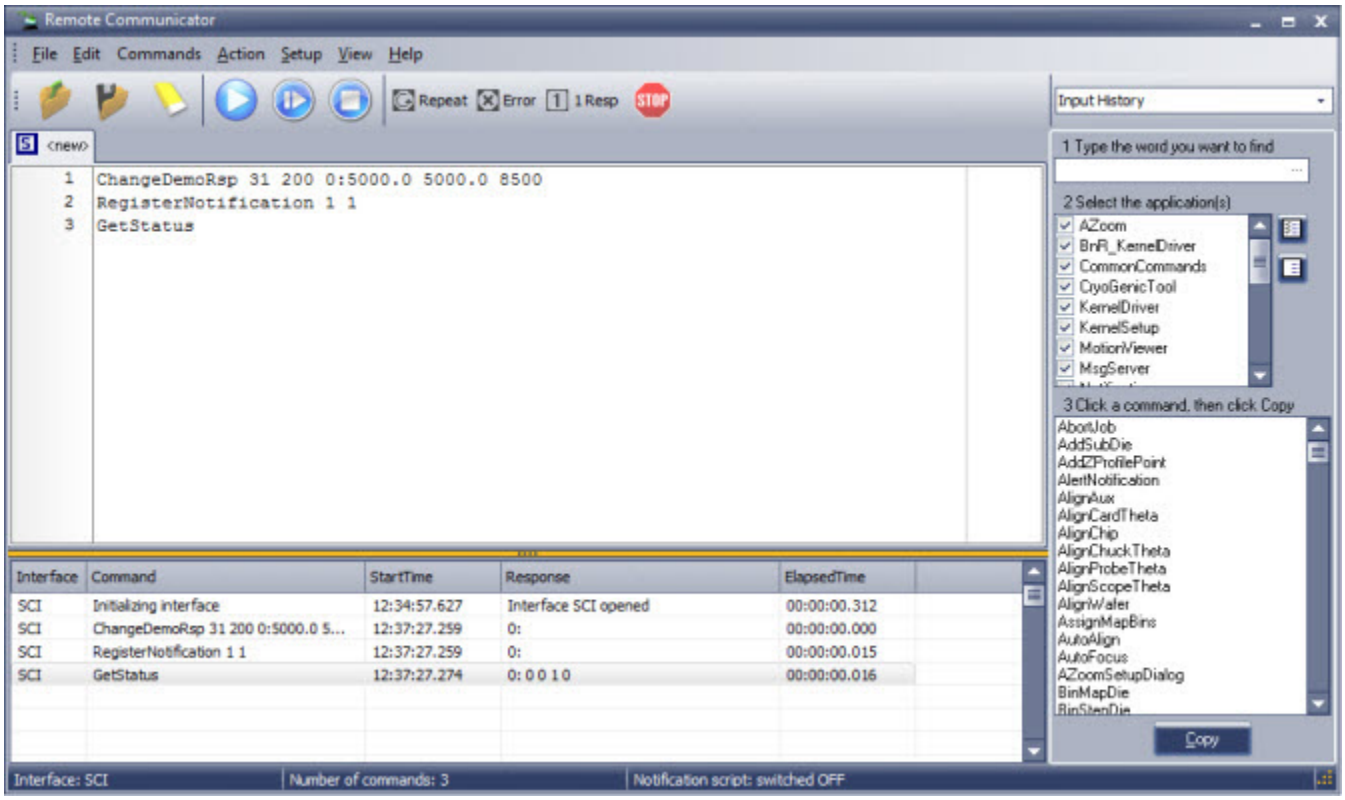
B Legacy Remote Communicator

Remote Communicator sends Velox remote commands for different interfaces, and generates sample sequences for probing in different script languages (Simple, Basic, Pascal).

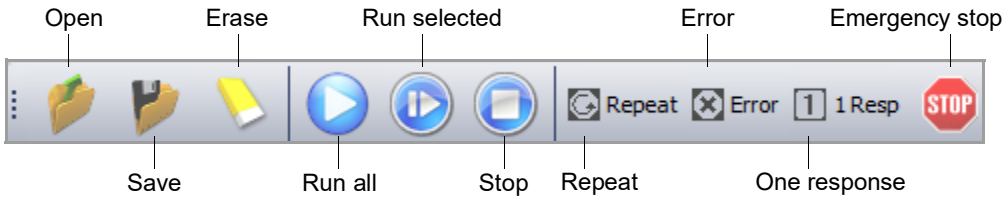
The Remote Communicator has been replaced by the Scripting Console in current versions of Velox. The Remote Communicator is available for customers who need legacy compatibility. It is not installed by default, but is available in the Additional Files folder of the DVD.

User Interface

The Remote Communicator interface lets you view SCI syntax, send commands, read prober responses, and test individual components of the system.



Toolbar



Open	Opens Window's common dialog box to open script files.
Save	Saves the active script with the current name.
Erase	Erases the logfile of the activity pane.
Run all	Runs all of the commands in the Script pane.
Run selected	Runs the selected command, then advances to the next command and pauses for the next user action.
Stop	Stop is used in conjunction with Start, to immediately stop the sending of commands.tr
Repeat	Repeat is used in conjunction with Start, to send the commands continually until Stop is clicked.
Error	Error is used to stop the sending of commands if an error response is received.
One response	One response displays the last response only, and works with Start or Step. This is useful if only the last response is of interest and you do not want to have to scroll down; and if a command is sent repeatedly and including display refresh time in the test is not desired.
Emergency stop	Emergency stop immediately sends the StopAllMovements command. This is useful to prevent personnel injury or machine damage in the event the user sees imminent risk.
Combo box	The commands combo box logs each command as it is sent to the prober. Selecting a command from the combo box appends it to the Script pane.

Log Window

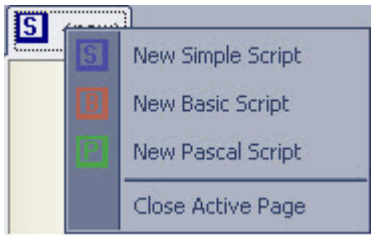
The Log window displays the commands that have been sent to the prober, and pertinent transmit information. Use Erase to clear the window.

Interface	Command	StartTime	Response	ElapsedTime
SCI	Initializing interface	12:34:57.627	Interface SCI opened	00:00:00.312

Interface	Indicates which interface was used to send the command.
Command	Typically, this is the same as the command in the Script pane.
Start Time	The time the command was sent.
Response	The response returned by the prober. All commands receive a response, minimally with 0, indicating no errors.
Elapsed Time	The time between sending the command and when the response was received.

Script Pane

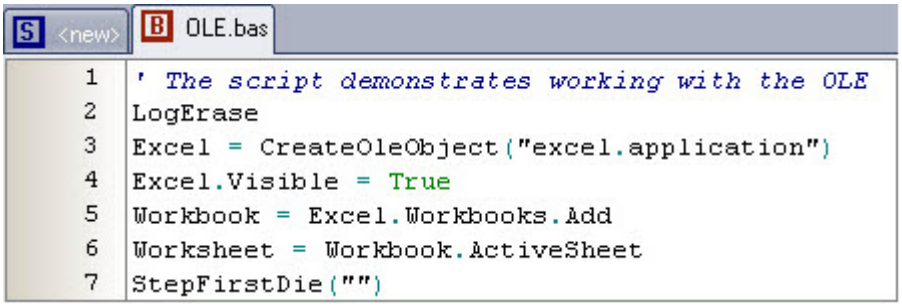
Commands can be entered using the Commands menu, the Recent Commands list, or directly with the keyboard. Commands can also be entered by opening a script file, and using basic text editing to make changes.

Commands menu	Commands entered with the Command menu include the command parameters with default values.
Keyboard entry	The default values can be changed using the keyboard.
Script pane	Single or multiple commands may be sent. When sending multiple commands, the commands are entered one per line, including any required or optional parameters. The commands entered in the pane may be saved in a script file for later use. More than one script can be opened at a time. Each script is located on a separate page.
S – Simple B – Basic P – Pascal	

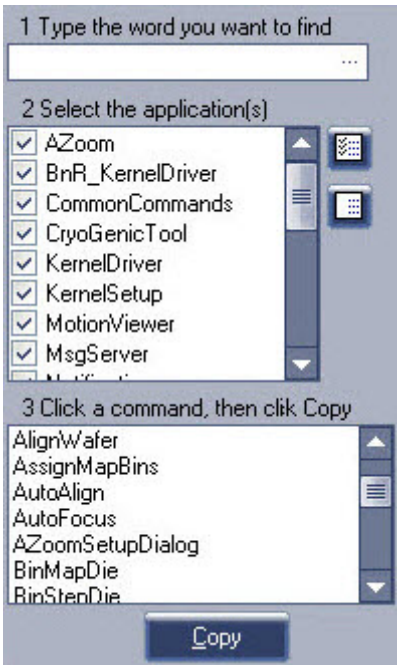
The program shows the active script's path on the main window's title bar.

A page can be closed or opened from the main menu: File\Close Script (shortcut Ctrl+F4) or from the popup menu (right mouse-click). If the program is started with the file names in the command line, it will open the files in the tabs. For example:

RComms.exe C:\Scripts\Hello.pas C:\Scripts\A_Nice_One.bas



Quick Command Window



1
To search for a command, enter the full or partial command name.

2
From the list, check applications that will filter associated remote commands.
The top button selects the applications that are checked.
The bottom button deselects the applications that are checked.

3
Displays available commands based on the specified criteria.

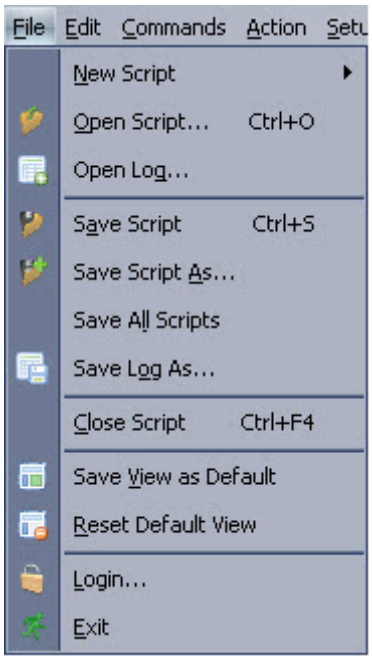
- Double-click any command, or
- Select a command, click Copy to paste the command to the script editor.

Status Bar

The status bar, located at the bottom of the main window, displays the current status of the Remote Communicator application.

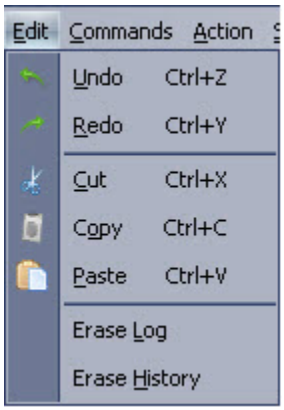
Interface	The current interface is displayed.
Number of commands	The number of the sent commands.
Notification script	Notification scripts, if any, are displayed. “Use notifications handling script” must be enabled (switched On) in Setup > Configuration > Miscellaneous.

File Menu



New Script	Opens the new page with a simple-style script. Opens the new page with a Basic script. Opens the new page with a Pascal script.
Open Script	Opens script files.
Open Log	Opens log files.
Save Script	Saves the active script with the current name.
Save Script As	Saves the script with new name.
Save All Scripts	Saves all opened scripts. If the script has no name, you are prompted for a new name.
Save Log As	Save log file to new name.
Close Script	Closes active script.
Save View as Default	Saves the names of the currently-opened scripts to the configuration.txt file. Next time, these scripts will open upon startup.
Reset Default View	Removes information about default view from the configuration file, so Remote Communicator will be opened with one blank script.
Login	Login with a different access level.
Exit	Closes the Remote Communicator.

Edit Menu



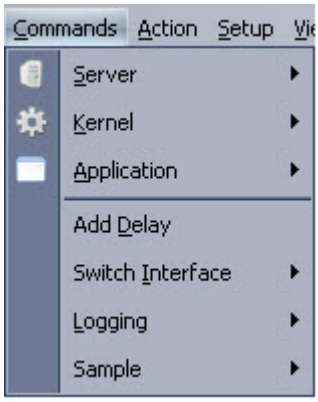
Undo	Undoes the last edit action.
Redo	Redoes the last Undo command.
Cut	Cuts the selected text and stores it in the clipboard.
Copy	Copies the selected text to the clipboard.
Paste	Pastes text stored in the clipboard.
Erase Log	Erases the logfile of the activity pane.
Erase History	Erases the history combo box.

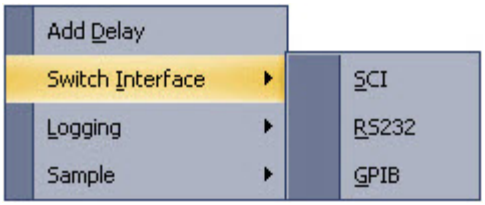
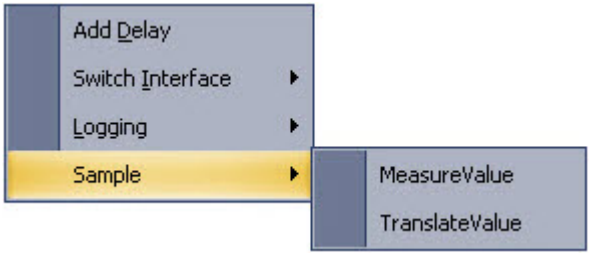
Commands Menu

Command Groupings

The Commands main menu includes additional menus that display all SCI commands. These menus follow the same logic as the SCI command set.

Selecting a command automatically appends the command to the Script pane, with placeholder arguments.



Commands menu	
Server	Server handles the queuing and routing commands and responses to the appropriate component.
Kernel	Kernel includes the basic commands required to operate the system. The Kernel menu divides the kernel command set into the system components that are affected by the command: Alignment, Chuck, Notifications, Probes, SW-Fence, Scope, Service, Setup, System, Theta, Tracer, Turret, ZProfile.
Application	Application includes higher level commands that combines kernel commands to duplicate the functionality of the Velox applications. Custom application development is faster and less error-prone when using the higher level commands.
Add Delay	Appends a time delay to the Script pane. Default is 100 milliseconds.
Switch Interface	
Logging	The logging commands are available for Pascal and Basic scripts.
Print to Log	Prints a string to the log.
On	Enables the logging for the Velox commands.
Off	Enables the logging for the Velox commands.
Erase	Erases the log.
Sample	

The Remote Communicator lets you attach additional libraries with the custom function. These libraries are located in the <Velox path>\RCLibs folder. The commands from the libraries are located below the Logging menu, grouped by the library name.

Command Groupings

To group the commands in the menu, comments are used in the commands.txt file:

;<New_Group_Name>

This tells the program to place the commands in a new submenu.

The program attaches the custom functions to the libraries. The library is a dll module that exports three functions. The libraries should be placed in the <Velox path>\RCLibs folder.

ConnectLib	void __stdcall ConnectLib(char *LibraryName, int *MethodsCount)
LibraryName	Name of the library
MethodsCount	Count of the methods implemented by the library

GetMethodOverview	bool __stdcall GetMethodOverview(int MethodIndex, TMethodOverview *Overview)
MethodItndex	Index of the method to describe, starts from zero
Overview	The method description

```
struct TMethodOverview
{
    char MethodName[80]; // method's name to show in the menu and use in the scripts
    bool ParamsRequired; // are incoming arguments required?
    int ReturnValuesCount; // count of the values in the response string
    char DefaultParams[100]; // default incoming arguments as string
    TCallMethodProc MethodHandler; // the function's handler
};
```

The handler is a function with the following prototype:

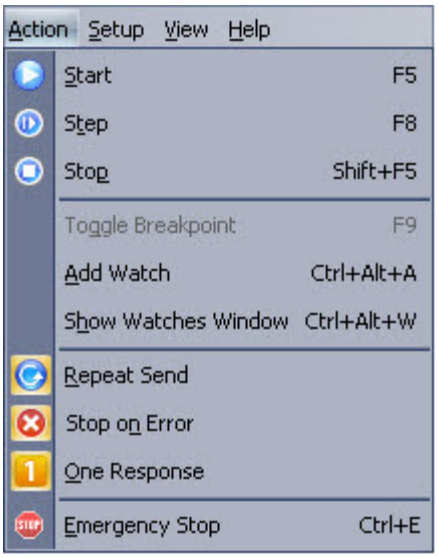
```
void __stdcall (*TCallMethodProc)(char* MethodName, char* Params, char* Response);
```

MethodName	Name of the called method
Params	Incoming function arguments
Response	The function fills it (if necessary) with a response string

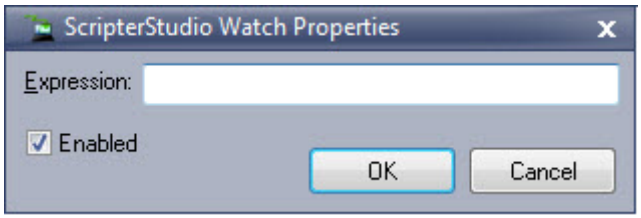
DisconnectLib	void __stdcall DisconnectLib()

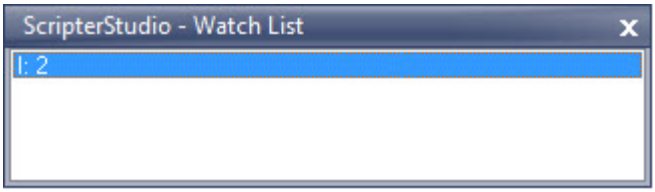
Action Menu

The Action menu defines how the system will respond to commands and display utilities. The most frequently used commands are also in the toolbar.



Start	Runs all of the commands in the Script pane.
Step	Runs the selected command, then advances to the next command and pauses for the next user action.
Stop	Stop is used in conjunction with Start, to immediately stop the sending of commands.
Toggle Breakpoint	Toggles a break point for the active script on the current line.
Add Watch	Adds a new variable to Watch List

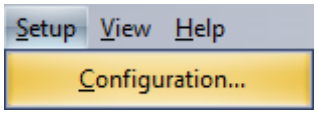


Show Watches Window	Watch List lets you monitor the values of the selected variables. 
Repeat Send	Repeat Send is used in conjunction with Start, to send the commands continually until Stop is clicked.
Stop on Error	Stop on Error is used in conjunction with Start, to stop the sending of commands if an error response is received.
One Response	Displays the last response only, and works with Start or Step. This is useful if only the last response is of interest and you do not want to have to scroll down; and if a command is sent repeatedly and including display refresh time in the test is not desired.
Emergency Stop	Sends the StopAllMovements command immediately. This is useful to prevent personnel injury or machine damage in the event the user sees imminent risk.

Setup Menu

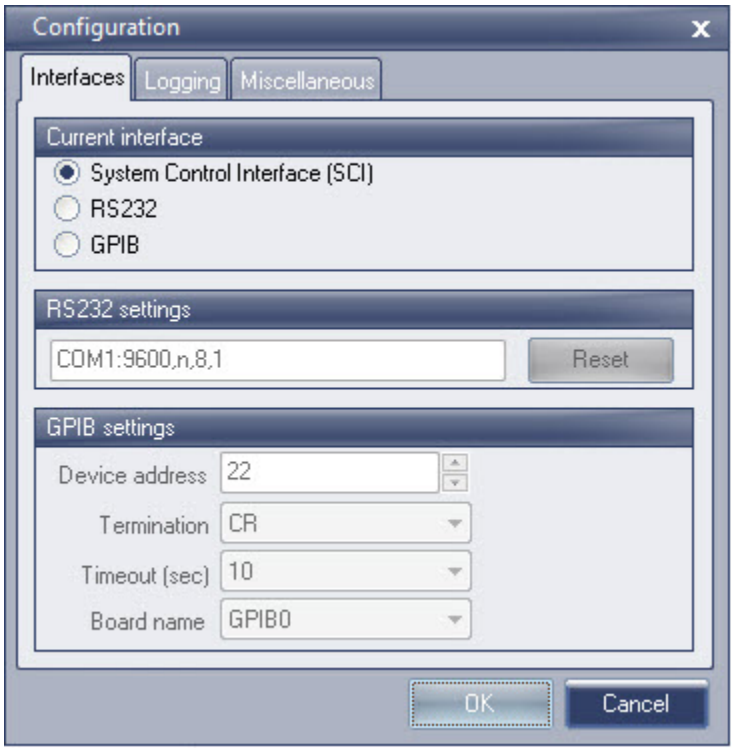
Configuration Interfaces
Configuration Logging
Configuration Miscellaneous
Notifications Handling Script

Setup functions include the configuring the interfaces, data logging, setting file extensions, project tracking, and notifications handling. Click Setup > Configuration to display the Configuration window.



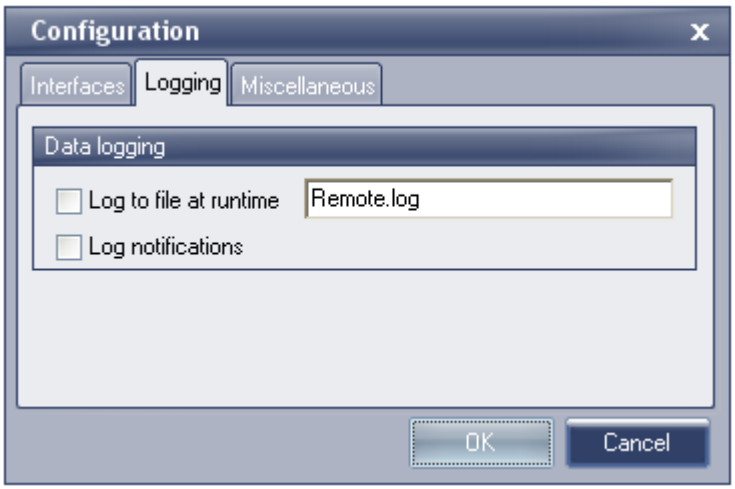
Configuration Interfaces

Interface pages let you select the active interface. Different options are available to edit, depending on the interface.



Configuration Logging

Data Logging writes all information that is recorded in the Activity pane to a text file.



Log to file at runtime

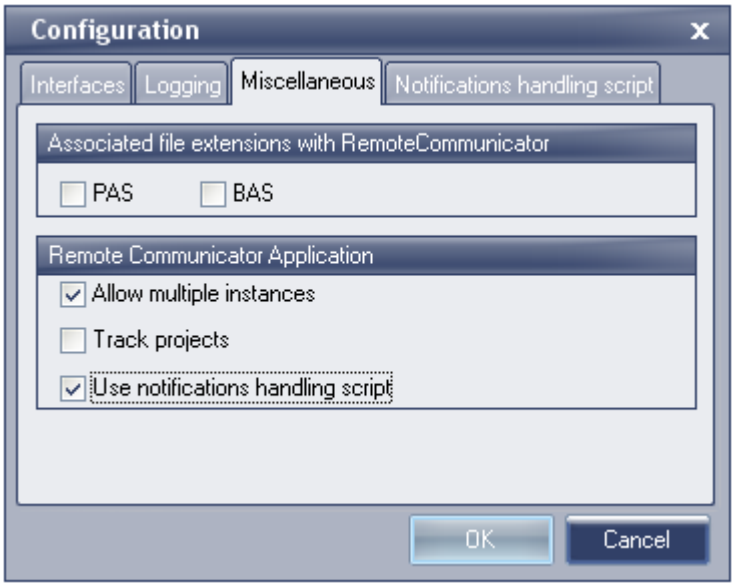
Enables writing to the log file specified in the file text box.

This option is disabled by default, therefore data logging must be enabled when Remote Communicator is started.

Log notifications

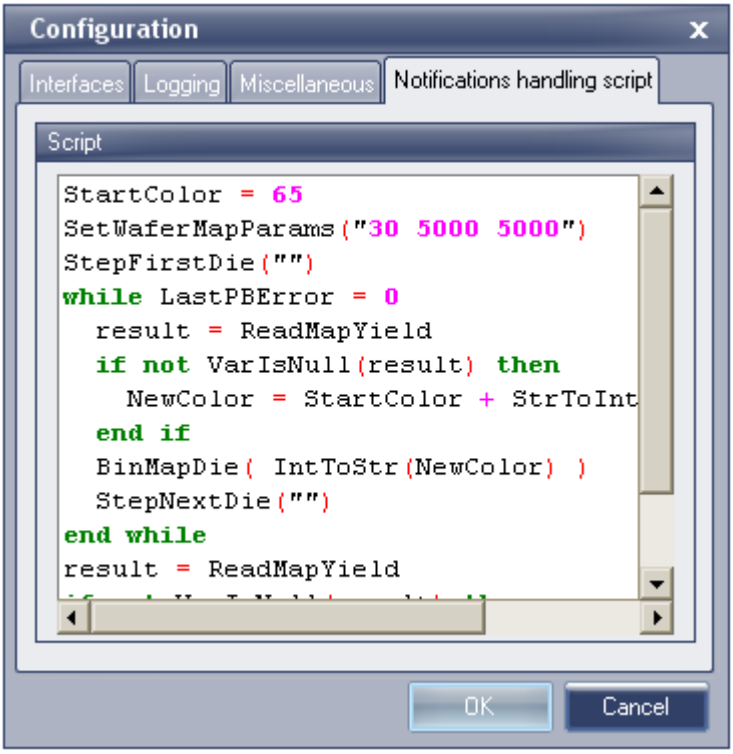
Displays the notification messages that occur between Velox applications.

Configuration Miscellaneous



Miscellaneous configuration settings
File extensions .pas – Pascal .bas – Basic
Allow multiple instances Enable to allow multiple instances of the application to run at the same time.
Track projects Instructs Communicator whether or not to use project files to save and read the program states.
Use notifications handling script Enables or disables the use of special script for the notifications handling.

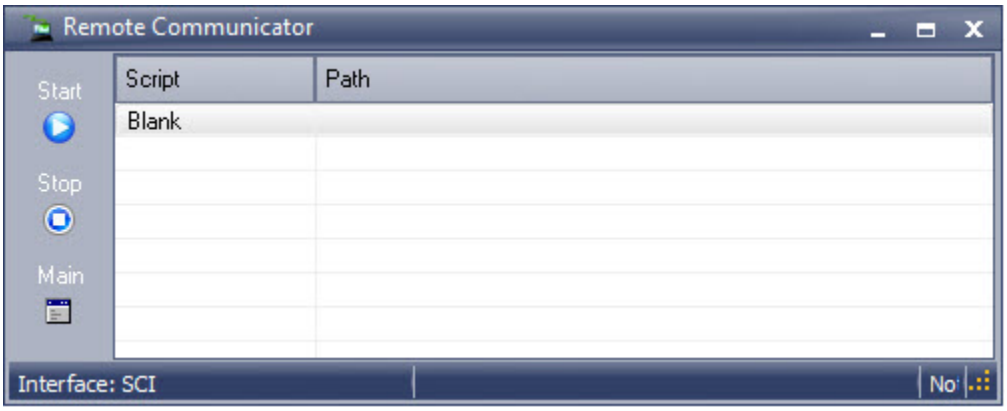
Notifications Handling Script



Available only if script is enabled.
Script to execute when a notification is received. (Velox commands are not allowed when using Pascal syntax.)
The following variables are predefined and available to use in the script: • CommandID is the number of the notification, (ex. AlertNotification ID is 202) • The number of parameters depends on the notification.

View Menu

Switch to MiniCommunicator

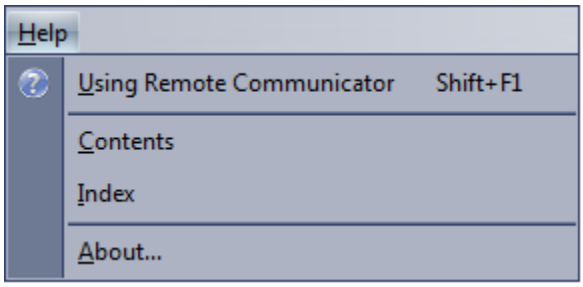


Script	The name of a currently-opened script. The active script is highlighted.

Path	The path of a currently-opened script.
Start	Starts execution of the script.
Stop	Stops execution of the script.
Main	Switches the Communicator to the normal view.
QuickCommand	This displays or hides the Quick Command.
Toolbar	This displays or hides the Toolbar.
Log Window	This displays or hides the Log Window.

Help Menu

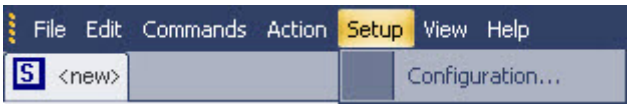
The Help menu provides access to the contents of the online help. To display product, version, and copyright information, click About.



Typical Setup

This procedure guides the user through a typical session of the Remote Communicator, from entering and running commands to saving in a script file.

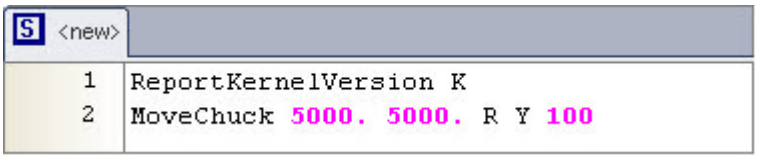
1. When the Remote Communicator is opened, the default interface setting is SCI. To change the interface, click Setup > Configuration > Interfaces.



2. To enter a command in the Script pane, click Commands > Kernel > System > ReportKernelVersion.
3. Click the Step button. The kernel version is displayed in the activity pane.

Interface	Command	StartTime	Response	ElapsedTime
SCI	ReportKernelVersion K	09:41:54.375	0: 0.000 DEMO MODE -> Version 0.000	00:00:00.031

4. To move the wafer chuck, click Commands > Kernel > Chuck > MoveChuck.
5. The command is entered into the Script pane with default x and y values of 5000 and 5000: These values may be edited using the keyboard.



6. Click to send the command.
   - All commands receive a response, either 0: if the command was handled without an error, or an error message.
   - If the command returns a response, such as the ReportKernelversion, the “0:” is followed by the command’s response.

Interface	Command	StartTime	Response	ElapsedTime
SCI	ReportKernelVersion K	09:41:54.375	0: 0.000 DEMO MODE -> Version 0.000	00:00:00.031
SCI	MoveChuck 5000, 5000, R Y 100	09:53:09.062	0:	00:00:00.750

7. To send all of the commands in the Script pane, click .
8. Commands may also be entered from the Recent Commands list, and directly from the keyboard.

9. Save the contents of the Script pane by clicking the Save button. Select the directory and enter the appropriate file name.

Keyboard Functions

File	Open Script	Ctrl + O
	Save Script	Ctrl + S
	Close Script	Ctrl + F4
Edit	Undo	Ctrl + Z
	Redo	Ctrl + Y
	Cut	Ctrl + X
	Copy	Ctrl + C
	Paste	Ctrl + V
Action	Start	F5
	Step	F8
	Stop	Shift + F5
	Toggle Breakpoint	F9
	Add Watch	Ctrl + Alt + A
	Show Watches Window	Ctrl + Alt + W
	Emergency Stop	Ctrl + E
View	Switch to Mini Communicator	Ctrl + O
	Quick Command Window	Ctrl + O
	Toolbar	Ctrl + O
	Log Window	Ctrl + O

Script Languages

This section describes the sample sequences for probing in different script languages (Simple, Basic, Pascal).

Simple Script Language

Send Velox Commands	All Velox commands are represented as functions with the same name. Commands can also be sent as command ID. Parameters should be separated by spaces. When using defaults, simply enter the command		
Example	Command	Command Function	Command ID
	BinMapDie 18	BinMapDie	602 18

Special Script Description commands can be added from Commands Menu > Add Delay > Switch Interface.

*Delay Pauses simple script execution for a specified number of milliseconds. Example: *Delay 1000
*SCI Instructs Remote Communicator to send the following commands using SCI interface. Example: *SCI
*RS232 Instructs Remote Communicator to send the following commands using RS232 interface. Example: *RS232
*GPIB Instructs Remote Communicator to send the following commands using GPIB interface. Example: *GPIB

Script Language (Pascal/Basic)

All Velox commands are represented as functions with the same name.

Send Velox Commands	Command function	Example
	If a command requires parameters, it should be passed as one string	`BinMapDie("18")`
	If a command requires an optional parameter and you want to use a default value, put empty string as an input parameter	`StepNextDie("")`
	If a function returns the response, it can be retrieved in the script as a variant array.	`MyResponse := ReadMapPosition`

MyResponse is a variant array of string, where each element is a returned value. If the response retrieval fails, then MyResponse variable is NULL. Use VarIsNull function to check correctness of the returned response.

Example:

```
result := ReadMapYield
if not VarIsNull(result) then
Print('Tested '+result[1]+' die(s)')
```

General Script Description (Pascal/Basic)

Overview	Pascal	Basic
	begin .. end constructor	**sub .. end and function .. end** declarations
	procedure and function declarations	**byref and dim** directives
	if .. then .. else constructor	**if .. then .. else .. end** constructor
	for .. to .. do .. step constructor	**for .. to .. step .. next** constructor
	while .. do constructor	**do .. while .. loop and do .. loop .. while** constructors
	repeat .. until constructor	**do .. until .. loop and do .. loop .. until** constructors
	try .. except and try .. finally blocks	**^ , * , / , and , + , - , or , <> , >=, <= , = , > , < , div , mod , xor , shl , shr** operators
	case statements	**try .. except and try .. finally** blocks
	array constructors (x:=[1, 2, 3];)	**select case .. end select** constructor
	^ , * , / , and , + , - , or , <> , >=, <= , = , > , < , div , mod , xor , shl , shr operators	**array constructors** (x:=[1, 2, 3];)
	access to object properties and methods (ObjectName.SubObject.Property)	**exit** statement
		access to object properties and methods (ObjectName.SubObject.Property)

Script Structure	Pascal	Basic
	Script structure is made of two major blocks: a) procedure (sub) and function declarations and b) main block. Both are optional, but at least one should be present in script. Some examples:	

	SCRIPT 1:
	procedure DoSomething;
	begin
	  CallSomething;
	end;
	
	begin
	  CallSomethingElse;
	end;
	
	SCRIPT 2:
	begin
	  CallSomethingElse;
	end;
	
	SCRIPT 3:
	function MyFunction;
	begin
	  result:='Ok!';
	end;
	
	SCRIPT 4:
	CallSomethingElse;
		SCRIPT 1:
		SUB DoSomething
		  CallSomething
		END SUB
		
		CallSomethingElse
		
		SCRIPT 2:
		CallSomethingElse
		
		SCRIPT 3:
		FUNCTION MyFunction
		  MyFunction = "Ok!"
		END FUNCTION

Identifiers	**Pascal**	**Basic**
	Identifier names in script (variable names, function and procedure names, etc.) follow the most common rules : should begin with a character (a..z or A..Z), or '_', and can be followed by alphanumeric chars or '_' char. Cannot contain any other character or spaces.	
	Valid Identifiers: VarName _Some V1A2 _____Some____ Invalid identifiers: 2Var My Name Some-more This,is,not,valid	Valid Identifiers: VarName _Some V1A2 _____Some____ Invalid identifiers: 2Var My Name Some-more This,is,not,valid
Assign statements	**Pascal**	**Basic**
	Assign statements (assign a value or expression result to a variable or object property) are built using ":=".	
	MyVar := 2; Button.Caption := 'This ' + 'is ok.';	MyVar = 2 Button.Caption = "This " + "is ok."
Character strings	**Pascal**	**Basic**
	Strings are declared in pascal using single quote (') character. Double quotes (") are not used. You can also use #nn to declare a character inside a string. There is no need to use '+' operator to add a character to a string. Examples: A := 'This is a text'; Str := 'Text '+'concat'; B := 'String with CR and LF char at the end '#13#10; C := 'String with '#33#34' characters in the middle';	Strings (sequence of characters) are declared in basic using double quote (") character. Examples: A = "This is a text" Str = "Text "+"concat"

Variables	Pascal	Basic

	There is no need to declare the type of variable in script. Thus, you declare variable just using var directive and its name. There is no need to declare variables if scripter property OptionExplicit is set to false. In this case, variables are implicitly declared. If you want to have more control over the script, set OptionExplicit property to true. This will raise a compile error if variable is used but not declared in script.
	SCRIPT 1:
	procedure Msg;
	var S;
	begin
	  S:='Hello world!';
	  ShowMessage(S);
	end;
	
	SCRIPT 2:
	var A;
	begin
	  A:=0;
	  A:=A+1;
	end;
	
	SCRIPT 3:
	var S;
	S:='Hello World!';
	ShowMessage(S);
		There is no need to declare the type of variable in script. Thus, you declare variable just using DIM directive and its name. There is no need to declare variables if scripter property OptionExplicit is set to false. In this case, variables are implicitly declared. If you want to have more control over the script, set OptionExplicit property to true. This will raise a compile error if variable is used but not declared in script.
		SCRIPT 1:
		SUB Msg
		  DIM S
		  S = "Hello world!"
		  ShowMessage(S)
		END SUB
		
		SCRIPT 2:
		DIM A
		A = 0
		A = A+1
		ShowMessage(A)
		
		Note that if script property OptionExplicit is set to false, then variable declarations are not necessary in any of scripts above.

Indexes	**Pascal**	**Basic**

		Strings, arrays and array properties can be indexed using "[" and "]" chars. For example, if Str is a string variable, the expression Str[3] returns the third character in the string denoted by Str, while Str[I + 1] returns the character immediately after the one indexed by I.
		`MyChar = MyStr[2]`
		`MyStr[1] = "A"`
		`MyArray[1,2] = 1530`
		`Lines.Strings[2] = "Some text"`

Arrays	**Pascal**	**Basic**
	Script supports array constructors and variant arrays. To construct an array, use "[" and "]" chars. You can construct multi-index array nesting array constructors. You can then access arrays using indexes. If array is multi-index, separate indexes using ",".	

	NewArray := [2,4,6,8];
	Num:=NewArray[1]; //Num receives "4"
	MultiArray := [['green','red','blue'] ,
	['apple','orange','lemon']];
	Str:=MultiArray[0,2]; //Str receives 'blue'
	MultiArray[1,1]:='new orange';
		NewArray = [2,4,6,8]
		Num = NewArray[1] //Num receives "4"
		MultiArray = [["green","red","blue"] ,
		["apple","orange","lemon"]]
		Str = MultiArray[0,2] //Str receives 'blue'
		MultiArray[1,1] = "new orange"

If statements	**Pascal**	**Basic**
	There are two forms of if statement: if...then and the if...then...else. Like normal pascal, if the if expression is true, the statement (or block) is executed. .	There are two forms of if statement: if...then..end if and the if...then...else..end if. Like normal basic, if the if expression is true, the statements are executed..

	`if J <> 0 then Result := I/J;`
	`if J = 0 then Exit else Result := I/J;`
	`if J <> 0 then`
	`begin`
	`  Result := I/J;`
	`  Count := Count + 1;`
	`end`
	`else`
	`  Done := True;`
		`IF J <> 0 THEN Result = I/J END IF`
		`IF J = 0 THEN Exit ELSE Result := I/J END IF`
		`IF J <> 0 THEN`
		`  Result = I/J`
		`  Count = Count + 1`
		`ELSE`
		`  Done = True`
		`END IF`

While statements	**Pascal**	**Basic**
	A while statement is used to repeat a statement or a block, while a control condition (expression) is evaluated as true. The control condition is evaluated before the statement. Hence, if the control condition is false at first iteration, the statement sequence is never executed. The while statement executes its constituent statement (or block) repeatedly, testing expression before each iteration. As long as expression returns True, execution continues.	A while statement is used to repeat statements, while a control condition (expression) is evaluated as true. The control condition is evaluated before the statements. Hence, if the control condition is false at first iteration, the statement sequence is never executed. The while statement executes its constituent statement repeatedly, testing expression before each iteration. As long as expression returns True, execution continues.

	`while Data[I] <> X do I := I + 1;`
	`while I > 0 do`
	`begin`
	`  if Odd(I) then Z := Z * X;`
	`  I := I div 2;`
	`  X := Sqr(X);`
	`end;`
	
	`while not Eof(InputFile) do`
	`begin`
	`  Readln(InputFile, Line);`
	`  Process(Line);`
	`end;`
		`WHILE (Data[I] <> X) I = I + 1 END WHILE`
		`WHILE (I > 0)`
		`  IF Odd(I) THEN Z = Z * X END IF`
		`  X = Sqr(X)`
		`END WHILE`
		
		`WHILE (not Eof(InputFile))`
		`  Readln(InputFile, Line)`
		`  Process(Line)`
		`END WHILE`

	Pascal	Basic

Repeat (loop) statements	The syntax of a repeat statement is repeat statement1; ...; statement; until expression where expression returns a Boolean value. The repeat statement executes its sequence of constituent statements continually, testing expression after each iteration.
	When expression returns True, the repeat statement terminates. The sequence is always executed at least once because expression is not evaluated until after the first iteration.
	repeat
	K := I mod J;
	I := J;
	J := K;
	until J = 0;
	
	repeat
	Write('Enter a value (0..9): ');
	Readln(I);
	until (I >= 0) and (I <= 9);
		Scripter support loop statements. Possible syntax:
		DO WHILE expr statements LOOP
		DO UNTIL expr statements LOOP
		DO statements LOOP WHILE expr
		DO statement LOOP UNTIL expr
		statements will be execute WHILE expr is true, or UNTIL expr is true. if expr is before statements, then the control condition will be tested before iteration. Otherwise, control condition will be tested after iteration.
		DO
		K = I mod J
		I = J
		J = K
		LOOP UNTIL J = 0
		
		DO UNTIL I >= 0
		Write("Enter a value (0..9): ")
		Readln(I)
		LOOP
		
		DO
		K = I mod J
		I = J
		J = K
		LOOP WHILE J <> 0
		
		DO WHILE I < 0
		Write("Enter a value (0..9): ")
		Readln(I)
		LOOP

For statements	**Pascal**	**Basic**

	Scripter support for statements with the following syntax: for counter := initialValue to finalValue do statement For statement set counter to initialValue, repeats execution of statement (or block) and increment value of counter until counter reaches finalValue.
	SCRIPT 1:
	for c:=1 to 10 do
	a:=a+c;
	
	SCRIPT 2:
	for i:=a to b do
	begin
	j:=i^2;
	sum:=sum+j;
	end;
		Scripter support for statements with the following syntax: FOR counter = initialValue TO finalValue STEP stepValue statements NEXT. For statement set counter to initialValue, repeats execution of statement until "next" and increment value of counter by stepValue, until counter reaches finalValue. Step part is optional, and if omitted stepValue is considered 1.
		SCRIPT 1:
		FOR c = 1 TO 10 STEP 2
		a = a + c
		NEXT
		
		SCRIPT 2:
		FOR I = a TO b
		j = i ^ 2
		sum = sum + j
		NEXT

	Pascal	Basic

Case statements	Scripter support case statements with the syntax:
	case selectorExpression of
	caseexpr1: statement1;
	...
	caseexprn: statementn;
	else
	elsestatement;
	end
	if selectorExpression matches the result of one of caseexprn expressions, the respective statement (or block) will be execute. Otherwise, elsestatement will be execute. Else part of case statement is optional. Different from Delphi, case statement in script doesn't need to use only ordinal values. You can use expressions of any type in both selector expression and case expression.
	case uppercase(Fruit) of
	'lime': ShowMessage('green');
	'orange': ShowMessage('orange');
	'apple': ShowMessage('red');
	else
	ShowMessage('black');
	end;
		Scripter support select case statements with syntax:
		SELECT CASE selectorExpression
		CASE caseexpr1
		statement1
		…
		CASE caseexprn
		statementn
		CASE ELSE
		elsestatement
		END SELECT
		if selectorExpression matches the result of one of caseexprn expressions, the respective statements will be execute. Otherwise, elsestatement will be executed. Else part of case statement is optional.
		
		SELECT CASE uppercase(Fruit)
		CASE "lime" ShowMessage("green")
		CASE "orange"
		ShowMessage("orange")
		CASE "apple" ShowMessage("red")
		CASE ELSE
		ShowMessage("black")
		END SELECT

	Pascal	Basic

Function and procedure (sub)declaration	Declaration of functions and procedures are similar to Object Pascal in Delphi, with the difference you don't specify variable types. Just like OP, to return function values, use implicated declared result variable. Parameters by reference can also be used, with the restriction mentioned: no need to specify variable types.
	procedure HelloWord;
	begin
	ShowMessage('Hello world!');
	end;
	
	procedure UpcaseMessage(Msg);
	begin
	ShowMessage(Uppercase(Msg));
	end;
	
	function TodayAsString;
	begin
	result:=DateToStr(Date);
	end;
	
	function Max(A,B);
	begin
	if A>B then
	result:=A
	else
	result:=B;
	end;
	
	procedure SwapValues(var A, B);
	Var Temp;
	begin
	Temp:=A;
	A:=B;
	B:=Temp;
	end;
		Declaration of functions and subs are similar to basic. In functions to return function values, use implicated declared variable which has the same name of the function. Parameters by reference can also be used, using BYREF directive.
		SUB HelloWord
		ShowMessage("Hello world!")
		END SUB
		
		SUB UpcaseMessage(Msg)
		ShowMessage(Uppercase(Msg))
		END SUB
		
		FUNCTION TodayAsString
		TodayAsString = DateToStr(Date)
		END FUNCTION
		
		FUNCTION Max(A,B)
		IF A>B THEN
		MAX = A
		ELSE
		MAX = B
		END IF
		END FUNCTION
		
		SUB SwapValues(BYREF A, B)
		DIM TEMP
		TEMP = A
		A = B
		B = TEMP
		END SUB

Special Script Description

Logging functions		
Print to Log	Prints a string to the log.	Pascal example: Print(‘print it!’) Basic example: Print(“print it!”)
LogOn	Enables the logging for the Velox commands.	Example: LogOn
LogOff	Disables the logging for the Velox commands.	Example: LogOff
LogErase	Erases the log.	Example: LogErase

Global Velox variables and functions		
LastError	Contains the last Velox command’s error code (integer).	
DataPath	Contains the path to the Velox folder (string).	
GetErrorDescription	Returns the error description string by the number	Example: GetErrorDescription(23)

Utility functions		
SwitchToSCI	Change the current interface.	
SwitchToRS232	Change the current interface.	
SwitchToGPIB	Change the current interface.	
OutText	The procedure sends the text through the currently selected interface.	OutText(string Text) OutText(‘ReportKernelVersion’)
EncodeString	Quotes a string.	Pascal example: EncodeString(‘C:\Temp\’) Basic example: EncodeString (“C:\Temp\”)
DecodeString	Removes a string quotation.	Pascal example: DecodeString(‘C:\\Temp\\’) Basic example: DecodeString (“C:\\Temp\\”)

InformUser	Shows a message box.
	function InformUser(Message: string; Caption: string; Flags: integer): Integer; where flags are a combination of the following constants:
	// Buttons
	mbABORTRETRYIGNORE
	mbOK
	mbOKCANCEL
	mbRETRYCANCEL
	mbYESNO
	mbYESNOCANCEL
	// Icons
	mbICONEXCLAMATION
	mbICONINFORMATION
	mbICONQUESTION
	mbICONSTOP
	If Flag is 0, combination of mbOK and mbICONINFORMATION is used.
	The return values are: IDABORT, IDCANCEL, IDIGNORE, IDNO, IDOK, IDRETRY, IDYES	

Global Script Variables

The communicator supports a set of variables accessible from all the scripts and from the remote commands.

SetFlag	Procedure sets the new value to the flag FlagValue: Integer);. Procedure SetFlag(FlagName: string);	SetFlag('WaferIsAligned', 0)
GetFlag	Function returns the value for the flag with FlagName name. Function GetFlag(FlagName: string): Integer;	if (GetFlag('ErrorOnLoad') = 1) then ….

Special Functions Descriptions

Pragma #alias	If the first line of a Pascal or Basic script includes #alias pragma this script becomes executable from the remote commands StartScript or DoScript. #alias <Name>	#alias LoadWafer

Internal Function Descriptions

Abs	Copy	EOF	Int	ReadLn	TimeToStr
AnsiCompareStr	Cos	Exp	IntToHex	Reset	Trim
AnsiCompareText	CreateOleObject	FilePos	IntToStr	Rewrite	TrimLeft
AnsiLowerCase	Date	FileSize	IsLeapYear	Round	TrimRight
AnsiUpperCase	DateTimeToStr	FloatToStr	IsValidIdent	Sin	Trunc
Append	DateToStr	Format	Length	Sqr	UpperCase
ArcTan	DayOfWeek	FormatDateTime	Ln	Sqrt	VarArrayCreate
Assigned	Dec	FormatFloat	Low	StrToDate	VarArrayHighBound
AssignFile	DecodeDate	Frac	LowerCase	StrToDateTime	VarArrayLowBound
ChDir	DecodeTime	GetActiveOleObject	Now	StrToFloat	VarIsNull
Chr	Delay	High	Odd	StrToInt	VarTo Str
CloseFile	Delete	Inc	Ord	StrToIntDef	Write
CompareStr	EncodeDate	IncMonth	Pos	StrToTime	WriteLn
CompareText	EncodeTime	Insert	Random	Time	

Abs

Description

Abs returns the absolute value of the argument X. X is an integer-type or real-type expression.

Declaration

function Abs(X);

Example

Pascal syntax

```
Result := Abs(Variable)
```

Basic syntax

```
Result = Abs(Variable)
```

AnsiCompareStr

Description

Compares strings based on the current locale with case sensitivity.

AnsiCompareStr compares S1 to S2, with case sensitivity. The compare operation is controlled by the current locale.

Condition Return Value

S1 > S2 > 0

S1 < S2 < 0

S1 = S2 = 0

Declaration

function AnsiCompareStr(const S1, S2: string): Integer;

Example

Pascal syntax

```
ResultStr := AnsiCompareStr(Str1, Str2)
```

Basic syntax

```
ResultStr = AnsiCompareStr(Str1, Str2)
```

AnsiCompareText

Description

Compares strings based on the current locale without case sensitivity.

AnsiCompareText compares S1 to S2, without case sensitivity. The compare operation is controlled by the current locale.

Condition	Return Value
S1 > S2	> 0
S1 < S2	< 0
S1 = S2	= 0

Declaration

function AnsiCompareText(const S1, S2: string): Integer;

Example

Pascal syntax

```
ResultStr := AnsiCompareText(Str1, Str2)
```

Basic syntax

```
ResultStr = AnsiCompareText(Str1, Str2)
```

AnsiLowerCase

Description

Returns a string that is a copy of the given string converted to lower case.

AnsiLowerCase returns a string that is a copy of the given string converted to lower case. The conversion uses the current locale.

Declaration

function AnsiLowerCase(const S: string): string;

Example

Pascal syntax

```
ResultStr := AnsiLowerCase(StrVar)
```

Basic syntax

```
ResultStr = AnsiLowerCase(StrVar)
```

AnsiUpperCase

Description

Converts a string to upper case.

AnsiUpperCase returns a string that is a copy of S, converted to upper case. The conversion uses the current locale.

Declaration

function AnsiUpperCase(const S: string): string;

Example

Pascal syntax

```
ResultStr := AnsiUpperCase(StrVar)
```

Basic syntax

```
ResultStr = AnsiUpperCase(StrVar)
```

Append

Description

Prepares an existing file for adding text to the end. Call Append to ensure that a file is opened with write-only access with the file pointer positioned at the end of the file. F is a text file variable and must be associated with an external file using AssignFile. If no external file of the given name exists, an error occurs. If F is already open, it is closed, then reopened. The current file position is set to the end of the file

Declaration

procedure Append(var F: Text);

Example

Pascal syntax

```
Var
F: Text;
AssignFile(F, 'C:\RC.log')
Append(F)
```

Basic syntax

```
dim F
AssignFile(F, "C:\RC.log")
Append(F)
```

ArcTan

Description

Calculates the arctangent of a given number. ArcTan returns the arctangent of X. X is a real-type expression that gives an angle in radians.

Declaration

function ArcTan(X: Extended): Extended;

Example

Pascal syntax

```
Result := ArcTan(Variable)
```

Basic syntax

```
Result = ArcTan(Variable)
```

Assigned

Description

Tests for a nil (unassigned) pointer or procedural variable. Use Assigned to determine whether the pointer or procedure referenced by P is nil. P must be a variable reference of a pointer or procedural type. Assigned(P) corresponds to the test P<> nil for a pointer variable, and @P <> nil for a procedural variable. Assigned returns false if P is nil, true otherwise.

Declaration

function Assigned(const P): Boolean;

Example

Pascal syntax

```
if Assigned(Pointer) then
begin
…
end
```

Basic syntax

```
IF Assigned(Pointer) THEN
…
END IF
```

AssignFile

Description

Associates the name of an external file with a file variable. Call AssignFile to initialize a file variable. F is a file variable of any file type. FileName is a string-type expression. After calling AssignFile, F is associated with the external file until F is closed. All further operations on the file variable F operate on the external file named by FileName.

Declaration

procedure AssignFile(var F; FileName: string);

Example

Pascal syntax

```
Var
F: Text;
AssignFile(F, 'C:\RC.log')
```

Basic syntax

```
dim F
AssignFile(F, "C:\RC.log")
```

ChDir

Description

Changes the current directory. ChDir changes the current directory to the path specified by S or P. If this operation fails, an exception is raised.

Declaration

procedure ChDir(const S: string);

Example

Pascal syntax

```
ChDir('C:\')
```

Basic syntax

```
ChDir("C:\")
```

Chr

Description

Returns the character for a specified ASCII value. Chr returns the character with the ordinal value (ASCII value) of the byte-type expression, X.

Declaration

function Chr(X: Byte): Char;

Example

Pascal syntax

```
CharVar := Chr(33)
```

Basic syntax

```
CharVar = Chr(33)
```

CloseFile

Description

Terminates the association between file variable and an external disk file.

Use the CloseFile procedure to terminate the association between a file variable and an external disk file. F is a file variable of any file type opened using Reset, Rewrite, or Append. The external file associated with F is completely updated and then closed, freeing the file handle for reuse.

Declaration

procedure CloseFile(var F);

Example

Pascal syntax

```
CloseFile(F)
```

Basic syntax

```
CloseFile(F)
```

CompareStr

Description

Compares two strings case-sensitivity.

CompareStr compares S1 to S2, with case-sensitivity. The return value is less than 0 if S1 is less than S2, 0 if S1 equals S2, or greater than 0 if S1 is greater than S2. The compare operation is based on the 8-bit ordinal value of each character and is not affected by the current locale.

Declaration

function CompareStr(S1, S2: string): Integer;

Example

Pascal syntax

```
Result := CompareStr(Str1, Str2)
```

Basic syntax

```
Result = CompareStr(Str1, Str2)
```

CompareText

Description

Compares two strings by ordinal value without case-sensitivity.

CompareStr compares S1 to S2, with case-sensitivity. The return value is less than 0 if S1 is less than S2, 0 if S1 equals S2, or greater than 0 if S1 is greater than S2. The compare operation is based on the 8-bit ordinal value of each character and is not affected by the current locale.

Declaration

function CompareText(const S1, S2: string): Integer;

Example

Pascal syntax

```
Result := CompareText(Str1, Str2)
```

Basic syntax

```
Result = CompareText(Str1, Str2)
```

Copy

Description

Returns a substring of a string.

S is an expression of a string. Index and Count are integer-type expressions. Copy returns a substring containing Count characters starting at S[Index].

Declaration

function Copy(S; Index, Count: Integer): string;

Example

Pascal syntax

```
ResultStr := Copy(Str, 1, 5)
```

Basic syntax

```
ResultStr = Copy(Str, 1, 5)
```

Cos

Description

Calculates the cosine of an angle. Cos returns the cosine of the angle X, X is a real-type expression that represents an angle in radians.

Declaration

function Cos(X: Extended): Extended;

Example

Pascal syntax

```
Result := Cos(Angle)
```

Basic syntax

```
Result = Cos(Angle)
```

CreateOleObject

Description

Instantiates an Automation object.

CreateOleObject creates a single uninitialized object of the class specified by the ClassName parameter. ClassName specifies the string representation of the Class ID (CLSID). CreateOleObject is used to create an object of a specified type when the CLSID is known, and when the object is on a local or in-proc server. Only objects that are not part of an aggregate are created using CreateOleObject.

Declaration

function CreateOleObject(const ClassName: string): IDispatch;

Example

Pascal syntax

```
ExcelObj := CreateOleObject('excel.application')
```

Basic syntax

```
ExcelObj = CreateOleObject(“excel.application”)
```

Date

Description

Returns the current date. Use Date to obtain the current local date as a TDateTime value. The time portion of the value is 0 (midnight).

Declaration

function Date: TDateTime;

Example

Pascal syntax

```
Day := DayOfWeek(Date)
```

Basic syntax

```
Day = DayOfWeek(Date)
```

DateTimeToStr

Description

DateTimeToStr converts the DateTime parameter, a TDateTime value, to a string.

Declaration

function DateTimeToStr(DateTime: TDateTime): string;

Example

Pascal syntax

```
StrResult := DateTimeToStr(Now)
```

Basic syntax

```
StrResult = DateTimeToStr(Now)
```

DateToStr

Description

Converts a TDateTime value to a string. Use DateToStr to obtain a string representation of a date value that can be used for display purposes.

Declaration

function DateToStr(Date: TDateTime): string;

Example

Pascal syntax

```
StrResult := DateToStr(Now)
```

Basic syntax

```
StrResult = DateToStr(Now)
```

DayOfWeek

Description

Returns the day of the week for a specified date.

DayOfWeek returns the day of the week of the specified date as an integer between 1 and 7, where Sunday is the first day of the week and Saturday is the seventh.

Declaration

function DayOfWeek(Date: TDateTime): Integer;

Example

Pascal syntax

```
Day := DayOfWeek(Now)
```

Basic syntax

```
Day = DayOfWeek(Now)
```

Dec

Description

Decrement (Dec) subtracts one from a variable.

Declaration

procedure Dec(var X);

Example

Pascal syntax

```
Dec(Variable)
```

Basic syntax

```
Dec(Variable)
```

DecodeDate

Description

Returns Year, Month, and Day values for a TDateTime value.

The DecodeDate procedure breaks the value specified as the Date parameter into Year, Month, and Day values. If the given TDateTime value has a negative (BC) year, the year, month, and day return parameters are all set to zero.

Declaration

procedure DecodeDate(Date: TDateTime; var Year, Month, Day: Word);

Example

Pascal syntax

```
DecodeDate(DateVar, Y, M, D)
```

Basic syntax

```
DecodeDate(DateVar, Y, M, D)
```

DecodeTime

Description

Breaks a TDateTime value into hours, minutes, seconds, and milliseconds.

DecodeTime breaks the object specified as the Time parameter into hours, minutes, seconds, and milliseconds.

Declaration

procedure DecodeTime(Time: TDateTime; var Hour, Min, Sec, MSec: Word);

Example

Pascal syntax

```
DecodeTime (TimeVar, H, M, S)
```

Basic syntax

```
DecodeTime (TimeVar, H, M, S)
```

Delay

Description

Pauses script execution for a specified number of milliseconds.

Declaration

procedure Delay procedure Delay(MiliSecValue : Integer);

Example

Pascal syntax

```
Delay(1803)
```

Basic syntax

```
Delay(1803)
```

Delete

Description

Removes a substring from a string.

Delete removes a substring of Count characters from string S starting with S[Index]. S is a string-type variable. Index and Count are integer-type expressions.

If index is larger than the length of the string or less than 1, no characters are deleted.

If count specifies more characters than remain starting at the index, Delete removes the rest of the string. If count is less than or equal to 0, no characters are deleted.

Declaration

procedure Delete(var S: string; Index, Count:Integer);

Example

Pascal syntax

```
Delete(Str, 2, 4)
```

Basic syntax

```
Delete(Str, 2, 4)
```

EncodeDate

Description

Returns a TDateTime value that represents a specified Year, Month, and Day.

EncodeDate returns a TDateTime value from the values specified as the Year, Month, and Day parameters.

- Year must be between 1 and 9999.
- Valid Month values are 1 through 12.
- Valid Day values are 1 through 28, 29, 30, or 31, depending on the Month value.

If the specified values are not within range, EncodeDate raises an exception.

Declaration

function EncodeDate(Year, Month, Day: Word): TDateTime;

Example

Pascal syntax

```
Result := EncodeDate(2012, 2, 18)
```

Basic syntax

```
Result = EncodeDate(2012, 2, 18)
```

EncodeTime

Description

Returns a TDateTime value for a specified Hour, Min, Sec, and MSec.

EncodeTime encodes the given hour, minute, second, and millisecond into a TDateTime value.

- Valid Hour values are 0 through 23.
- Valid Min and Sec values are 0 through 59.
- Valid MSec values are 0 through 999.

If the specified values are not within range, EncodeTime raises an exception.

Declaration

function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;

Example

Pascal syntax

```
Result := EncodeTime(21, 15, 0)
```

Basic syntax

```
Result = EncodeTime(21, 15, 0)
```

EOF

Description

EOF tests whether the current file position is the end-of-file. F is a file variable that has been opened for reading. Eof(F) returns true if the current file position is beyond the last character of the file or if the file is empty; otherwise, Eof(F) returns false.

Declaration

function Eof(var F): Boolean;

Example

Pascal syntax

```
If EOF(F) then
CloseFile(F);
```

Basic syntax

```
IF EOF(F) THEN
CloseFile(F)
END IF
```

Exp

Description

Returns the exponential of X. Exp returns the value of e raised to the power of X, where e is the base of the natural logarithms.

Declaration

function Exp(X: Real): Real;

Example

Pascal syntax

```
Result := Exp(Variable)
```

Basic syntax

```
Result = Exp(Variable)
```

FilePos

Description

Returns the current file position. Use FilePos on the file variable of an open file to determine the current position in the file. If the current position in the file is at the very beginning, FilePos returns 0. Otherwise, FilePos returns the byte offset from the beginning of the current position. FilePos will not work on a file that is not open.

Declaration

function FilePos(var F): Longint;

Example

Pascal syntax

```
FPos := FilePos(F)
```

Basic syntax

```
FPos = FilePos(F)
```

FileSize

Description

Returns number of records in a file. Call FileSize to determine the size of the file specified by the file variable F. To use FileSize, the file must be open. If the file is empty, FileSize(F) returns 0.

Declaration

function FileSize(var F): Integer;

Example

Pascal syntax

```
FSize := FileSize(F)
```

Basic syntax

```
FSize = FileSize(F)
```

FloatToStr

Description

Converts a floating point value to a string.

FloatToStr converts the floating-point value given by Value to its string representation. The conversion uses general number format with 15 significant digits.

Declaration

function FloatToStr(Value: Extended): string;

Example

Pascal syntax

```
StrVar := FloatToStr(FloatVar)
```

Basic syntax

```
StrVar = FloatToStr(FloatVar)
```

Format

Description

Returns a formatted string assembled from a format string and an array of arguments.

Formats the series of arguments in an open (untyped) array. Format is the format string. Args is an array of arguments to apply to the format specifiers in Format.

Format specifiers have the following form: "%" [index ":"] ["-"] [width] ["." prec] type

The following table summarizes the possible values for type:

d = Decimal – Argument must be an integer value
u = Unsigned decimal – Similar to 'd' but no sign is output
s = Scientific – Argument must be a floating-point value
f = Fixed – Argumeant must be a floating-point value
g = General – Argument must be a floating-point value
n = Number – Argument must be a floating-point value
m = Money – Argument must be a floating-point value
p = Pointer – Argument must be a character or a string
s = String – Argument must be a character or a string
x = Hexadecimal – Argument must be an integer value

Declaration

function Format(const Format: string; const Args: array of const): string;

Example

Pascal syntax

```
StrVar := Format('value: %d', [IntVar])
```

Basic syntax

```
StrVar = Format("value: %d", [IntVar])
```

FormatDateTime

Description

Formats a TDateTime value. Call FormatDateTime formats the TDateTime value given by DateTime using the format given by Format.

Declaration

function FormatDateTime(const Format: string; DateTime: TDateTime): string;

Example

Pascal syntax

```
Str := FormatDateTime('"Today is " dddd', Now)
```

Basic syntax

```
Str = FormatDateTime("'Today is ' dddd", Now)
```

FormatFloat

Description

Formats a floating point value. FormatFloat formats the floating-point value given by Value using the format string given by Format. The following format specifiers are supported in the format string:

0 – Digit place holder
– Digit placeholder
. – Decimal point
, – Thousand separator
E+ – Scientific notation
'xx'/"xx" – Characters enclosed in single or double quotes are output as-is, and do not affect formatting.
; – Separates sections for positive, negative, and zero numbers in the format string.

Declaration

function FormatFloat(const Format: string; Value: Extended): string;

Example

Pascal syntax

```
StrVar := FormatFloat('"Number" #.00', FloatValue)
```

Basic syntax

```
StrVar = FormatFloat('Number #.00', FloatValue)
```

Format	Result string			
	1234	-1234	0.5	0
0	1234	-1234	1	0
0.00	1234.00	-1234.00	0.50	0.00
#.##	1234	-1234	.5	
#,##0.00	1,234.00	-1,234.00	0.50	0.00
#,##0.00;(#,##0.00)	1,234.00	(1,234.00)	0.50	0.00
#,##0.00;;Zero	1,234.00	-1,234.00	0.50	Zero
0.000E+00	1.234E+03	-1.234E+03	5.000E-01	0.000E+00
#.###E-0	1.234E3	-1.234E3	5E-1	0E0

Frac

Description

Returns the fractional part of a real number. Frac function returns the fractional part of the argument X. X is a real-type expression. The result is the fractional part of X; that is, Frac(X) = X - Int(X).

Declaration

function Frac(X: Extended): Extended;

Example

Pascal syntax

```
Result := Frac(Variable)
```

Basic syntax

```
Result = Frac(Variable)
```

GetActiveOleObject

Description

Retrieves a reference to an IDispatch interface to a currently running, registered COM object.

GetActiveOleObject returns a reference to the IDispatch interface of the active object for the given class indicated by ClassName. ClassName is the programmatic ID for the class of the active object from the OLE running object table. GetActiveOleObject is used to access the OLE running object table. To add an object to this table, use the RegisterActiveObject API.

If either the the class object or the IDispatch interface are not successfully returned, an exception is raised.

Declaration

function GetActiveOleObject(const ClassName: string): IDispatch;

Example

Pascal syntax

```
ExcelObj := GetActiveOleObject ('excel.application')
```

Basic syntax

```
ExcelObj = CreateOleObject("excel.application")
```

High

Description

Returns the highest value in the range of an argument.

Call High to obtain the upper limit of an Ordinal or Array.

Declaration

function High(X);

Example

Pascal syntax

```
Limit := High(Variable)
```

Basic syntax

```
Limit = High(Variable)
```

Inc

Description

Increment (Inc) adds one to the variable X.

Declaration

procedure Inc(var X);

Example

Pascal syntax

```
Inc(Variable)
```

Basic syntax

```
Inc(Variable)
```

IncMonth

Description

Returns a date shifted by a specified number of months. IncMonth returns the value of the Date parameter, incremented by NumberOfMonths months. NumberOfMonths can be negative, to return a date N months previous.

Declaration

function IncMonth(const Date: TDateTime; NumberOfMonths: Integer): TDateTime;

Example

Pascal syntax

```
NewDate := IncMonth(Now, 12)
```

Basic syntax

```
NewDate = IncMonth(Now, 12)
```

Insert	Description	Inserts a substring into a string beginning at a specified point. Insert merges Source into S at the position S[index]. If Index is less than 1, it is mapped to a 1. If it is past the end of the string, it is set to the length of the string, turning the operation into an append. If the Source parameter is an empty string, Insert does nothing.	
	Declaration	procedure Insert(Source: string; var S: string; Index: Integer);	
	Example	Pascal syntax	`Insert(SourceStr, DestStr, 5)`
		Basic syntax	`Insert(SourceStr, DestStr, 5)`

Int

Description

Returns integer part of a real number. Int returns the integer part of X; that is, X rounded toward zero. X is a real-type expression.

Declaration

function Int(X: Extended): Extended;

Example

Pascal syntax

```
Result := Int(Variable)
```

Basic syntax

```
Result = Int(Variable)
```

IntToHex

Description

Returns the hex representation of an integer.

IntToHex converts a number into a string containing the number's hexadecimal (base 16) representation. Value is the number to convert. Digits indicates the minimum number of hexadecimal digits to return.

Declaration

function IntToHex(Value: Integer; Digits: Integer): string;

Example

Pascal syntax

```
HexStr := IntToHex(IntVar, 4)
```

Basic syntax

```
HexStr = IntToHex(IntVar, 4)
```

IntToStr

Description

Converts an integer to a string.

IntToStr converts an integer into a string containing the decimal representation of that number.

Declaration

function IntToStr(Value: Integer): string;

Example

Pascal syntax

```
StrValue := IntToStr(IntValue)
```

Basic syntax

```
StrValue = IntToStr(IntValue)
```

IsLeapYear

Description

Indicates whether a specified year is a leap year. Call IsLeapYear to determine whether the year specified by the Year parameter is a leap year. Year specifies the calendar year.

Declaration

function IsLeapYear(Year: Word): Boolean;

Example

Pascal syntax

```
if IsLeapYear (1979) then
begin
…
end
```

Basic syntax

```
IF IsLeapYear (1979) THEN
…
END IF
```

IsValidIdent

Description

Tests for a valid identifier.

IsValidIdent returns true if the given string is a valid identifier. An identifier is defined as a character from the set ['A'..'Z', 'a'..'z', '_'] followed by zero or more characters from the set ['A'..'Z', 'a'..'z', '0..'9', '_'].

Declaration

function IsValidIdent(const Ident: string): Boolean;

Example

Pascal syntax

```
Result := IsValidIdent('BoolVar')
```

Basic syntax

```
Result = IsValidIdent('BoolVar')
```

Length

Description

Returns the number of characters in a string or elements in an array.

Length returns the number of characters actually used in the string or the number of elements in the array.

Declaration

function Length(S): Integer;

Example

Pascal syntax

```
StrLen := Length(StrVar)
```

Basic syntax

```
StrLen = Length(StrVar)
```

Ln

Description

Returns the natural log of a real expression. Int returns the integer part of X; that is, X rounded toward zero. X is a real-type expression.

Declaration

function Ln(X: Real): Real;

Example

Pascal syntax

```
Result := Ln(Variable)
```

Basic syntax

```
Result = Ln(Variable)
```

Low

Description

Returns the lowest value in a range. Call Low to obtain the lowest value or first element of an Ordinal or Array.

Declaration

function Low(X);

Example

Pascal syntax

```
Result := Low(Variable)
```

Basic syntax

```
Result = Low(Variable)
```

LowerCase

Description

Returns a copy of a string in lowercase.

LowerCase returns a string with the same text as the string passed in S, but with all letters converted to lowercase. The conversion affects only 7-bit ASCII characters between 'A' and 'Z'. To convert 8-bit international characters, use AnsiLowerCase.

Declaration

function LowerCase(S: string): string;

Example

Pascal syntax

```
LowerCase(StrVar)
```

Basic syntax

```
LowerCase(StrVar)
```

Now

Description

Returns the current date and time, corresponding to the sum of the value returned by the global Date and Time functions.

Declaration

function Now: TDateTime;

Example

Pascal syntax

```
DecodeTime (Now, H, M, S)
```

Basic syntax

```
DecodeTime (Now, H, M, S)
```

Odd

Description

Returns true if argument is an odd number. Odd tests if the argument is an odd number. It returns true if X is an odd number, false if X is even.

Declaration

function Odd(X: Longint): Boolean;

Example

Pascal syntax

```
IsOdd := Odd(Variable)
```

Basic syntax

```
IsOdd = Odd(Variable)
```

Ord

Description

Returns the ordinal value of an ordinal-type expression. X is an ordinal-type expression. The result is the ordinal position of X; its type is the smallest standard integer type that can hold all values of X's type.

Declaration

function Ord(X);

Example

Pascal syntax

```
Result := Ord(Variable)
```

Basic syntax

```
Result = Ord(Variable)
```

Pos

Description

Returns the index value of the first character in a specified substring that occurs in a given string.

Pos searches for Substr within S and returns an integer value that is the index of the first character of Substr within S. Pos is case-sensitive. If Substr is not found, Pos returns zero.

Declaration

function Pos(Substr: string; S: string): Integer;

Example

Pascal syntax

```
StrPos := Pos('.', StrVar)
```

Basic syntax

```
StrPos = Pos('.', StrVar)
```

Random

Description

Generates random numbers.

Random returns a real-type random number within the range 0 <= X < 1.

Declaration

function Random;

Example

Pascal syntax

```
RndValue := Random
```

Basic syntax

```
RndValue = Random
```

ReadLn

Description

Reads a line of text from a file.

ReadLn procedure reads a line of text and then skips to the next line of the file.

Declaration

procedure ReadLn(var F: Text) : string;

Example

Pascal syntax

```
ReadLn(F, Str)
```

Basic syntax

```
ReadLn(F, Str)
```

Reset

Description

Opens an existing file. Call AssignFile to initialize a file variable. Reset opens the existing external file with the name assigned to F. An error results if no existing external file of the given name exists or if the file can't be opened. The current file position is set to the beginning of the file.

Declaration

procedure Reset(var F: File);

Example

Pascal syntax

```
Var
F: Text;
AssignFile(F, 'C:\RC.log')
Reset(F)
```

Basic syntax

```
dim F
AssignFile(F, "C:\RC.log")
Reset(F)
```

Rewrite

Description

Creates a new file and opens it.

Rewrite creates a new external file with the name assigned to F.

Declaration

procedure Rewrite(var F: File);

Example

Pascal syntax

```
Var
F: Text;
AssignFile(F, 'C:\RC.log')
Rewrite(F)
```

Basic syntax

```
dim F
AssignFile(F, "C:\RC.log")
Rewrite(F)
```

Round

Description

Returns the value of X rounded to the nearest whole number. The Round function rounds a real-type value to an integer-type value. X is a real-type expression. Round returns an Int64 value that is the value of X rounded to the nearest whole number. If X is exactly halfway between two whole numbers, the result is always the even number. This method of rounding is often called "Banker's Rounding". If the rounded value of X is not within the Int64 range, a runtime error is generated.

Declaration

function Round(X: Extended): Int64;

Example

Pascal syntax

```
RoundedVar := Round(Variable)
```

Basic syntax

```
RoundedVar = Round(Variable)
```

Sin

Description

Returns the sine of the angle in radians.
. X is a real-type expression. Sin returns the sine of the angle X in radians.

Declaration

function Sin(X: Extended): Extended;

Example

Pascal syntax

```
Result := Sin(Angle)
```

Basic syntax

```
Result = Sin(Angle)
```

Sqr

Description

Returns the square of a number. X is an ordinal-type expression.
Sqr function returns the square of the argument.

Declaration

function Sqr(X: Extended): Extended;

Example

Pascal syntax

```
Result := Sqr(Variable)
```

Basic syntax

```
Result = Sqr(Variable)
```

Sqrt

Description

Returns the square root of X.

X is a floating-point expression. The result is the square root of X.

Declaration

function Sqrt(X: Extended): Extended;

Example

Pascal syntax

```
Result := Sqrt(Variable)
```

Basic syntax

```
Result = Sqrt(Variable)
```

StrToDate

Description

Converts a string to a TDateTime value. Call StrToDate to parse a string that specifies a date. If S does not contain a valid date, StrToDate raises an exception.

Declaration

function StrToDate(const S: string): TDateTime;

Example

Pascal syntax

```
D := StrToDate('18.02.2012')
```

Basic syntax

```
D = StrToDate("18.02.2012")
```

StrToDateTime

Description

Converts a string to a TDateTime value. Call StrToDateTime to parse a string that specifies a date and time value. If S does not contain a valid date, StrToDateTime raises an exception.

Declaration

function StrToDateTime(const S: string): TDateTime;

Example

Pascal syntax

```
D := StrToDateTime('25.05.2001 18:24:31')
```

Basic syntax

```
D = StrToDateTime("25.05.2001 18:24:31")
```

StrToFloat

Description

Converts a given string to a floating-point value. Use StrToFloat to convert a string, S, to a floating-point value. S must consist of an optional sign (+ or -), a string of digits with an optional decimal point, and an optional mantissa. The mantissa consists of 'E' or 'e' followed by an optional sign (+ or -) and a whole number. Leading and trailing blanks are ignored.

Declaration

function StrToFloat(const S: string): Extended

Example

Pascal syntax

```
ExtVar := StrToFloat(StrVar)
```

Basic syntax

```
ExtVar = StrToFloat(StrVar)
```

StrToInt

Description

Converts a string that represents an integer (decimal or hex notation) to a number.

StrToInt converts the string S, which represents an integer-type number in either decimal or hexadecimal notation, into a number. If S does not represent a valid number, StrToInt raises an exception.

Declaration

function StrToFloat(const S: string): Extended

Example

Pascal syntax

```
IntVar := StrToInt(StrVar)
```

Basic syntax

```
IntVar = StrToInt(StrVar)
```

StrToIntDef

Description

Converts a string that represents an integer (decimal or hex notation) to a number with error default.

StrToIntDef converts the string S, which represents an integer-type number in either decimal or hexadecimal notation, into a number. If S does not represent a valid number, StrToIntDef returns Default.

Declaration

function StrToIntDef(const S: string; const Default: Integer): Integer;

Example

Pascal syntax

```
IntVar := StrToIntDef(StrVar, 0)
```

Basic syntax

```
IntVar = StrToIntDef(StrVar, 0)
```

StrToTime

Description

Converts a string to a TDateTime value. Call StrToTime to parse a string that specifies a time value. If S does not contain a valid time, StrToTime raises an exception.

Declaration

function StrToTime(const S: string): TDateTime;

Example

Pascal syntax

```
D := StrToTime('17:52:23')
```

Basic syntax

```
D = StrToTime("17:52:23")
```

Time

Description

Returns the current time. Time returns the current time as a TDateTime value.

Declaration

function Time: TDateTime;

Example

Pascal syntax

```
DecodeTime (Time, H, M, S)
```

Basic syntax

```
DecodeTime (Time, H, M, S)
```

TimeToStr

Description

Returns a string that represents a TDateTime value. TimeToStr converts the Time parameter, a TDateTime value, to a string.

Declaration

function TimeToStr(Time: TDateTime): string;

Example

Pascal syntax

```
StrResult := TimeToStr(Now)
```

Basic syntax

```
StrResult = TimeToStr(Now)
```

Trim

Description

Trims leading and trailing spaces and control characters from a string. Trim removes leading and trailing spaces and control characters from the given string S.

Declaration

function Trim(const S: string): string;

Example

Pascal syntax

```
Result := Trim(StrVar)
```

Basic syntax

```
Result = Trim(StrVar)
```

TrimLeft

Description

Trims leading spaces and control characters from a string. TrimLeft returns a copy of the string S with leading spaces and control characters removed.

Declaration

function TrimLeft(const S: string): string;

Example

Pascal syntax

```
Result := TrimLeft(StrVar)
```

Basic syntax

```
Result = TrimLeft(StrVar)
```

TrimRight

Description

Trims trailing spaces and control characters from a string. TrimRight returns a copy of the string S with trailing spaces and control characters removed.

Declaration

function TrimRight(const S: string): string;

Example

Pascal syntax

```
Result := TrimRight(StrVar)
```

Basic syntax

```
Result = TrimRight(StrVar)
```

Trunc

Description

Truncates a real number to an integer. The Trunc function truncates a real-type value to an integer-type value. X is a real-type expression. Trunc returns an Int64 value that is the value of X rounded toward zero.

If the truncated value of X is not within the Int64 range, a runtime error is generated.

Declaration

function Trunc(X: Extended): Int64;

Example

Pascal syntax

```
Result := Trunc(Variable)
```

Basic syntax

```
Result = Trunc(Variable)
```

UpperCase

Description

Returns a copy of a string in uppercase.

UpperCase returns a copy of the string S, with the same text but with all 7-bit ASCII characters between 'a' and 'z' converted to uppercase. To convert 8-bit international characters, use AnsiUpperCase.

Declaration

function UpperCase(S: string): string;

Example

Pascal syntax

```
UpperCase(StrVar)
```

Basic syntax

```
Result = Trunc(Variable)
UpperCase(StrVar)
```

VarArrayCreate

Description

VarArrayCreate creates a variant array with the bounds given by Bounds and the element type given by VarType.

The Bounds parameter must contain an even number of values, where each pair of values specifies the upper and lower bounds of one dimension of the array.

Possible TVarType constants are (not declared):

varNull = $0001;
varSmallint = $0002;
varInteger = $0003;
varSingle = $0004;
varDouble = $0005;
varCurrency = $0006;
varDate = $0007;
varOleStr = $0008;
varDispatch = $0009;
varError = $000A;
varBoolean = $000B;
varUnknown = $000D;
varShortInt = $0010;
varByte = $0011;
varWord = $0012;
varLongWord = $0013;
varInt64 = $0014;

Declaration

function VarArrayCreate(const Bounds: array of Integer; VarType: TVarType): Variant;

Example

Pascal syntax

```
A := VarArrayCreate([0, 4], varVariant)
```

Basic syntax

```
A = VarArrayCreate([0, 4], varVariant)
```

VarArrayHighBound

Description

Returns high bound for a dimension in a variant array.

VarArrayHighBound returns the high bound of the given dimension in the given variant array. The Dim parameter should be 1 for the leftmost dimension, 2 for the second leftmost dimension, and so on. An exception is raised if the variant given by A is not an array, or if the dimension specified by Dim does not exist.

Declaration

function VarArrayHighBound(const A: Variant; Dim: Integer): Integer;

Example

Pascal syntax

```
bound := VarArrayHighBound(VArray, 0)
```

Basic syntax

```
bound = VarArrayHighBound(VArray, 0)
```

VarArrayLowBound

Description

Returns the low bound of a dimension in a variant array.

VarArrayLowBound returns the low bound of the given dimension in the given variant array. The Dim parameter should be 1 for the leftmost dimension, 2 for the second leftmost dimension, and so on. An exception is raised if the variant given by A is not an array, or if the dimension specified by Dim does not exist.

Declaration

function VarArrayLowBound(const A: Variant; Dim: Integer): Integer;

Example

Pascal syntax

```
bound := VarArrayLowBound(VArray, 0)
```

Basic syntax

```
bound = VarArrayLowBound(VArray, 0)
```

VarIsNull

Description

Indicates whether the specified variant is Null. VarIsNull returns true if the given variant contains the value Null. If the variant contains any other value, the function result is false.

Note: *Do not confuse a Null variant with an unassigned variant. A Null variant is still assigned, but has the value Null. Unlike unassigned variants, Null variants can be used in expressions and can be converted to other types of variants.*

Declaration

function VarIsNull(const V: Variant): Boolean;

Example

Pascal syntax

```
If VarIsNull(Variable) then
Begin
…
End
```

Basic syntax

```
IF VarIsNull(Variable) THEN
…
END IF
```

VarTo Str

Description

Converts a variant's value to a string.

VarToStr converts the data in the variant V to a string and returns the result. If the variant has a null value, VarToStr returns an empty string.

Declaration

function VarToStr(const V: Variant): string;

Example

Pascal syntax

```
StrVar := VarToStr(VariantVar)
```

Basic syntax

```
StrVar = VarToStr(VariantVar)
```

Write

Description

Writes a string to a file.

Declaration

procedure Write(var F: File; S: string);

Example

Pascal syntax

```
Write(F, Str)
```

Basic syntax

```
Write(F, Str)
```

WriteLn

Description

Writes to a text file and adds an end-of-line marker. WriteLn is an extension to the Write procedure, as it is defined for text files. After executing Write, WriteLn writes an end-of-line marker (line feed or carriage return/line feed) to the file.

Declaration

procedure WriteLn(var F: Text; S: string);

Example

Pascal syntax

```
WriteLn(F, Str)
```

Basic syntax

```
WriteLn(F, Str)
```

Appendix

C Legacy Communication Protocols

The Dynamic Data Exchange Server application (DDEInterface.exe) allows you to write automated test programs to control your probe station. The DDE Server accepts commands from other Windows-based applications, sends them to the probe station by way of the Command Server, and returns responses.

DDE Server Communications

The DDE application is based on the DDE messaging system built into Windows. The concepts and terms presented here are common to all DDE applications.

Communication with the DDE Server follows the client/server model. Essentially, two Windows-based programs carry on a “conversation” by posting messages to each other. These two programs are known as the “server” and the “client”. A server, such as DDE Server, is the program that has access to data (or system hardware) that the other program needs. A client is the program that requests and receives data from the server.

The client program initiates DDE conversations. The client broadcasts a message to all running Windows-based programs that indicates the type of information it needs. The server that has the information, in this case DDE Server, responds and the conversation begins.

Conversations between a DDE Client and Server are opened with a pair of names that identify the application and topic that the client program needs to converse about. The application and topic names are always the same (shown below).

The item name is the specific data item or command that the client program is to control.

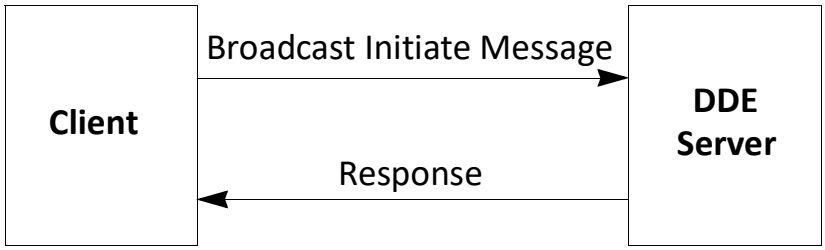
Application Name:	EDMAIN
Topic Name:	CMI Commands
Item Name:	Any of the remote commands that are supported by Velox. For example: `:move:load 2`

Windows supports three basic types of DDE interprocess conversations: cold link, hot link and warm link. DDE Server uses only the cold link (or execution).

A DDE conversation begins when the client calls `DdeConnect()` which broadcasts a message to initiate the conversation.

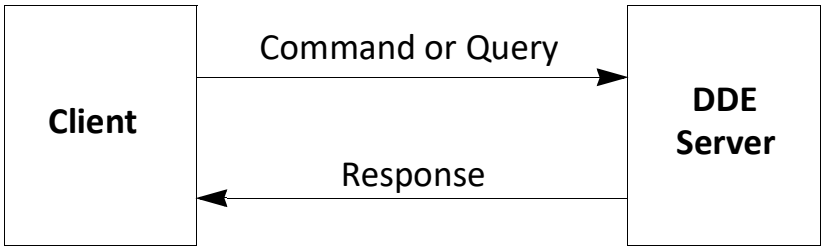
The server responds to the client with an acknowledge message.

Client initiates conversation



The client sends commands to the DDE Server using the `DdeClientTransaction()` `Application Program Interface (API)`.

DDE Server always returns responses



For each client call, the DDE Server uses the `DdeClientTransaction()` return value as a pointer to a status string which contains one of the following values:

- A string containing `"COMPLETE"`, indicating that the transaction executed without error.
- A string containing the answer to a query made by the client. The format of the string depends on the nature of the requested data.
- A string containing an error message. The first character of any error-message string is the symbol @. The error message, a new line character combination, and the command string that generated the error condition follow this.

The conversation ends when the client posts a disconnect messages to the server. The DDE Server disconnects without returning an acknowledge message.

Client terminates DDE Server conversations

DDE Programming Examples

Two code fragments demonstrating the use of the DDE Server are shown in this section:

- Example: C
- Example: Visual Basic

Example: C

The following code fragment shows how to send a command to the DDE Server. It uses the Windows API DdeClientTransaction() to send a **:move:relative** command.

```
// The following code fragment uses Windows DDEML APIs
// to send a move command to the DDE Server
// and receives a status string.
//
// Variables and initial values.
HCONV hConv = NULL;
DWORD idInst = 0;
HSZ hszService = NULL;
HSZ hszTopic = NULL;
HSZ hszCmd = NULL;
BOOL err = FALSE;
HDDEDATA transResult = NULL;
DWORD dwResult = 0;
char result[300];
// Need long timeout value to allow the station to move and
// respond.
const long TIMEOUT = 60000;
// Initialize the DDEML environment. Create an Instance
// that is used in many other calls to DDE.
if(DMLERR_NO_ERROR != DdeInitialize(&idInst,MyDDECallBack,
APPCMD_CLIENTONLY, 0))
err = true;

// Create String handles for the server name, topic and item.
if(!err)
{
hszService = DdeCreateStringHandle(idInst, "EDMAIN",
CP_WINANSI);
hszTopic = DdeCreateStringHandle(idInst, "CMI Commands",
CP_WINANSI);
hszCmd = DdeCreateStringHandle(idInst,
":MOVE:REL 2 100 100 NONE",
CP_WINANSI);
err = (hszService == NULL || hszTopic == NULL ||
hszCmd == NULL);
}

// Connect to the DDE Server. (Open a conversation).
// Captain Picard would say, "Open a channel Mr. Wharf".
if(!err)
{
hConv = DdeConnect(idInst, hszService, hszTopic, NULL);
err = hConv == NULL;
if(err)
```

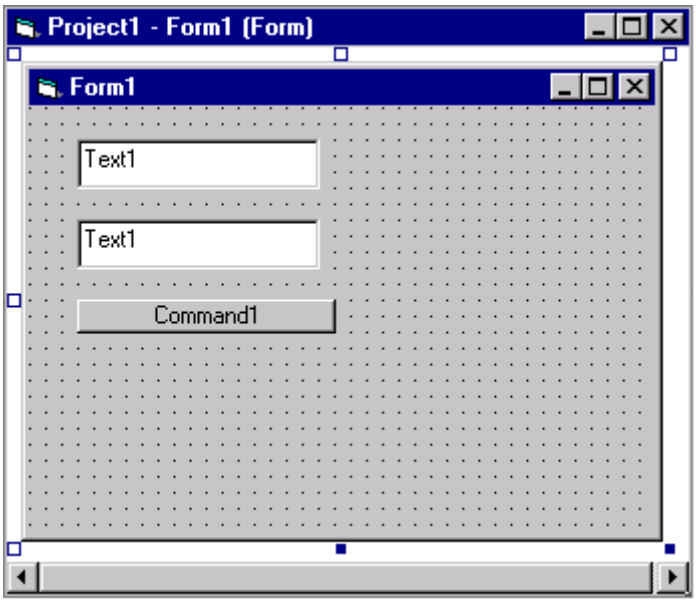
```
MessageBox(NULL, "Unable to make DDE connection.\n"
"Make sure Velox is running.",
"DDE CLIENT ERROR", MB_ICONSTOP);
}

// Send the command string to the server.
if (!err)
{
transResult = DdeClientTransaction(NULL, 0, hConv, hszCmd,
CF_TEXT, XTYP_REQUEST, TIMEOUT, NULL);

// Read the result string. TransResult will be a
// valid data handle if the client transaction above
// was successful, and NULL if it was not. This must be
// checked since calls to DdeGetData with a NULL handle
// cause GPF's.

if(transResult)
DdeGetData(transResult, (LPBYTE)result,
sizeof(result), 0);
// Display the result string.
MessageBox(NULL, result,
"RESULT", MB_ICONINFORMATION);
}
// Close the conversation.
if (hConv != NULL)
DdeDisconnect( hConv );
// Delete the string handles.
if ((hszService != NULL) & (idInst != NULL))
DdeFreeStringHandle( idInst, hszService );
if ((hszTopic != NULL) & (idInst != NULL))
DdeFreeStringHandle( idInst, hszTopic );
if ((hszCmd != NULL) & (idInst != NULL))
DdeFreeStringHandle( idInst, hszCmd );
// Clear out the DDEML environment.
if (idInst != NULL)
DdeUninitialize(idInst);
..

// Since we are only doing requests from Velox, we don't
// expect to get callbacks to this routine. Nevertheless,
// it is necessary to create a routine with no action.
HDDEDATA CALLBACK MyDDECallBack( UINT wType,
UINT wFmt, HCONV HConv, HSZ dataHandle1, HSZ dataHandle2,
   HDDEDATA data, DWORD myword1, DWORD myword2)
{
    return NULL;
}
```

Example: Visual Basic

The following subroutine uses Visual Basic 6.0 to create a message box that you can use to send commands directly to the DDE Server.

```
' Simple Visual Basic program that communicates
' with Velox via DDE.
```

```
'
' Steps:
' 1. Start Visual Basic 6.0.
' 2. Create a new Standard EXE program. VB will
'    make a blank form.
' 3. Add a text box. In the properties window
'    change the name of this text box to CmdText.
' 4. Add a second text box. Change its name
'    to RespText.
' 5. Add a button. VB will create the button
'    with the default name Command1.
' 6. Double click on the button (in design mode).
'    VB will open the code for the button. Enter
'    the routine as shown below.
' 7. Press F5 (start run).
' 8. Enter a command string in the first edit
'    box on the form (CmdText). For example, enter
'    "MOVE:REL 2 100 100 NONE" without the quote marks.
' 9. Make sure that the DDE server is running.
' 10. Click the Command1 button. Velox will execute
'     the command and return a response string
'     which will be shown in the RespText box.

Option Explicit
Private Sub Command1_Click()
    RespText.LinkItem = CmdText.Text
    RespText.LinkTopic = "EDMAIN|CMI COMMANDS"
    RespText.LinkMode = vbLinkManual
    RespText.LinkRequest
    RespText.LinkMode = vbLinkNone

EndSub
```



Index

Symbols
*CLS *21*
*Delay *366*
*ESE *22*
*ESE? *22*
*ESR? *22*
*GPIB *366*
*IDN? *22*, *23*
*RS232 *366*
*RST *23*
*SCI *366*
*SRE *23*
*SRE? *23*
*STB? *24*
*TST? *24*

A
AbortJob *102*
AddSubDie *137*
AlertNotification *235*
AlignAux *91*
AlignAuxOffAxis *91*
AlignCardTheta *157*
AlignChip *92*
AlignChuckTheta *158*
AlignProbeTheta *158*
AlignScopeTheta *159*
AlignWafer *92*
ASCII strings, Kernel *238*
AssignMapBins *126*
AutoAlign *92*
AutomationSearchCurrentDie *93*
AZoomSetupDialog *76*

B
BinMapDie *126*
BinStepDie *127*
BinSubDie *137*

C
CancelTTLTest *100*
CleanProbeTip *200*
ClearAllBins *152*
CloseCommunicator *80*
ConfirmRecipe *102*
ConvertToAlphas *110*

D
DeleteAllSubDie *138*
DeleteSubDie *138*
DeleteSubDie2 *138*
DetectWaferHeight *93*
DockCassette *102*
DoInkerRun *127*
DoScript *80*
DoTTLTest *100*
DoWaferProfiling *154*

E
EchoData *212*
EnableEdgeSensor *227*
EnableHeaterHoldMode *229*
EnableHeaterStandby *229*
EnableMotorQuiet *212*
EnableOffset *226*
EnableOverlay *84*
ExecuteCleaningSequence *67*

F
FindFeature *94*
FindFocus *95*
FindWaferCenter *95*

G
GetActiveLayer *111*
GetAlignmentMode *68*
GetAutomationActive *84*
GetAutomationTemperatureStatus *84*
GetAuxSiteCount *200*
GetAuxSiteName *200*
GetAxisReverse *213*
GetAZoomLens *76*
GetBackSideMode *213*
GetBinCode *152*
GetBinTableSize *152*
GetCameraLight *81*
GetCameraView *82*
GetCassetteStatus *103*
GetCleaningParams *201*
GetClusterDieStatus *146*
GetClusterInfo *147*
GetClusterParams *147*
GetConstantContactModeStatus *85*
GetCurrentBin *111*
GetDarkMode *214*

GetDewPointTemp *230*
GetDieDataAsColRow *127*
GetDieDataAsNum *128*
GetDieInfo *128*
GetDieLabel *129*
GetDieLabelAsNum *129*
GetDieMapResult *138*
GetDieMapResultAsNum *139*
GetDieRefPoint *139*
GetDieResult *129*
GetDieResultAsNum *130*
GetDieStatus *130*
GetHeaterSoak *230*
GetHeaterTemp *231*
GetJobList *103*
GetJobParams *103*
GetLoginData *68*
GetLotID *111*
GetMapDims *111*
GetMapHome *112*
GetMapOrientation *112*
GetMapRoute *113*
GetNumSelectedClusters *148*
GetNumSelectedDies *130*
GetOffsetInfo *227*
GetPreMappedDieInfo *113*
GetProbingStatus *104*
GetProductID *114*
GetProjectFile *68*
GetReAlignStatus *88*
GetReAlignTemperatureStatus *88*
GetRectMapParams *114*
GetRunStatus *80*
GetSelectedClusterCoords *148*
GetSelectedDieCoords *131*
GetSoftwarePath *69*
GetSoftwareStop *214*
GetStageLock *214*
GetStatus *69*
GetSubDieData *139*
GetSubDieDataAsColRow *140*
GetSubDieDataAsNum *141*
GetSubDieLabel *141*
GetSubDieLabelAsNum *141*
GetSubDieStatus *142*
GetTargetTemp *231*
GetTTLLines *100*
GetTTLStatus *100*
GetWaferCenter *95*
GetWaferID *115*
GetWaferInfo *115*
GetWaferMapMode *115*
GetWaferMapParams *116*
GetWaferMapParams2 *116*
GetWaferNum *117*
GetWaferProfileOptions *154*
GetWaferProfilingStatus *155*
GetWaferTestAngle *117*
GoToWaferHome *117*

H

HeatChuck *232*

I

InitChuck *161*
InitProbe *174*
InitScope *190*
InkDevice *215*
IsAppRegistered *70*

J

JobStatus *104*

K

KernelConnectionStatus *72*

L

Legacy Remote Communicator scripts
    Basic script .BAS files *62*
    conversion to native Python functions *62*
    file conversion *62*
    library functions *63*
LoadWafer *105*
LocateHomeDie *96*
LoginDialog *70*

M

Message Server
    applications list *30*
    log filter configuration *31*
    user interface *30*
MoveAuxSite *201*
MoveAZoomFocus *77*
MoveAZoomVelocity *77*
MoveChuck *162*
MoveChuckAlign *162*
MoveChuckContact *163*
MoveChuckIndex *163*
MoveChuckLift *163*
MoveChuckLoad *164*
MoveChuckSeparation *164*
MoveChuckSubsite *164*
MoveChuckTransfer *165*
MoveChuckVelocity *165*
MoveChuckZ *165*
MoveChuckZSafe *166*
MoveProbe *175*

MoveProbeAlign *176*
MoveProbeContact *177*
MoveProbeIndex *177*
MoveProbeLift *178*
MoveProbeSeparation *179*
MoveProbeVelocity *179*
MoveProbeZ *180*
MoveScope *191*
MoveScopeAlign *191*
MoveScopeFocus *191*
MoveScopeIndex *192*
MoveScopeLift *192*
MoveScopeSeparation *192*
MoveScopeSilo *193*
MoveScopeVelocity *193*
MoveScopeZ *193*
MoveTheta *209*
MoveThetaVelocity *209*

N

NewAccessLevel *72*
NewProjectFile *72*
NewWaferMap *117*
NextWafer *88*
Notifications, AlertNotification *235*
Notifications, Server *72*

O

OCGetZoomLevel *77*
OCLightOn *78*
OCLightVal *78*
OCSetZoomLevel *78*
OpenBinCodeTable *153*
OpenProject *70*
OpenProjectDialog *70*
OpenWaferMap *117*
OrientProbe *181*

P

PreMapWafer *96*
ProbeCardOCS *96*
ProbeToPadAlign *97*
ProceedJob *106*
ProceedProbing *106*
ProcessWafer *106*
PSProgressChanged *73*
PSStateChanged *73*
Python Scripting Language
- adding 3rd party Python libraries *61*
- IntelliSense and automatic code completion *59*
- introduction to programming *60*
- library modules *54*
- overview *52*
- SCI commands *55*

Q

QueryCassetteID *107*
QueryWaferID *107*
QueryWaferInfo *107*

R

Read2DMatrixCode *97*
ReadAuxHeights *202*
ReadAuxIndex *202*
ReadAuxPosition *203*
ReadAuxStatus *203*
ReadAZoomFocus *79*
ReadBarCode *97*
ReadCardTheta *159*
ReadChuckHeights *166*
ReadChuckIndex *167*
ReadChuckPosition *167*
ReadChuckStatus *168*
ReadChuckThermoScale *169*
ReadChuckThermoValue *169*
ReadChuckTheta *160*
ReadClusterPosition *148*
ReadContactCount *215*
ReadCurrentLens *211*
ReadMapPosition *118*
ReadMapPosition2 *119*
ReadMapYield *119*
ReadOcrString *98*
ReadProbeHeights *182*
ReadProbeIndex *183*
ReadProbePosition *183*
ReadProberStatus *216*
ReadProbeSetup *217*
ReadProbeStatus *184*
ReadProbeTheta *160*
ReadScopeHeights *194*
ReadScopeIndex *194*
ReadScopePosition *195*
ReadScopeSilo *195*
ReadScopeSiloCount *196*
ReadScopeStatus *196*
ReadScopeTheta *161*
ReadSensor *218*
ReadSystemStatus *219*
ReadTemperatureChuckStatus *232*
ReadThetaPosition *210*
ReadThetaStatus *210*
ReadTypedSensor *220*
ReadWaferStatus *221*
ReAlign *89*
RegisterProberAppChange *74*
Remote Communicator

internal functions *375–383*
keyboard functions *365*
script languages *366–374*
setup procedure *364*
user interface *353–364*
ReportKernelVersion *222*
ReportSoftwareVersion *71*
ResetAutomation *85*
ResetCleaningPosition *204*
ResetContactCount *222*
RS232 command protocol *5*
RS232 settings *8*
RS232 user interface *7*

S

SaveBinCodeTable *153*
SaveMapFile *120*
SaveProject *71*
SaveProjectAsDialog *71*
SaveProjectFile *74*
SCPI commands
Alignment *256*
Automation *258*
Capture Video *260*
Move *261*
Probeplan *261*
Vision *350*
Wafer Map *261*
SCPI commands summary *250*
SCPI device IDs *256*
Scripting Console
Command log *50*
Edit menu, advanced *42*
Edit menu, basic *42*
editor window *46*
Error list *51*
Event log *50*
File menu *41*
GPIB serial communications *64*
legacy Remote Communicator scripts *61*
Options menu, Logging *45*
Options menu, User Interface *45*
overview *36*
Run/Debug menu *43*
Script output *51*
Simple Sequential scripting language *51*
user interface *40*
Variables *51*
SearchChuckContact *170*
SelectAZoomLens *79*
SelectLens *211*
SetActiveLayer *120*
SetAlignmentMode *71*
SetAutomationActive *85*
SetAuxHeight *205*
SetAuxHome *205*
SetAuxIndex *206*
SetAuxMode *206*
SetAuxSiteCount *207*
SetAuxSiteName *207*
SetAuxSiteType *207*
SetAuxThetaHome *208*
SetAZoomLight *79*
SetBeaconStatus *222*
SetBinCode *153*
SetBinTableSize *154*
SetCameraLight *82*
SetCameraQuiet *83*
SetCameraView *83*
SetChuckHeight *171*
SetChuckHome *171*
SetChuckIndex *172*
SetChuckMode *172*
SetChuckThermoScale *173*
SetChuckThermoValue *173*
SetChuckVacuum *223*
SetCleaningParams *208*
SetClusterDieStatus *149*
SetClusterParams *149*
SetConstantContactMode *86*
SetCurrentBin *120*
SetDarkMode *223*
SetDieDataAsColRow *131*
SetDieDataAsNum *131*
SetDieLabel *132*
SetDieLabelasNum *132*
SetDieMapResult *142*
SetDieMapResultAsNum *143*
SetDieRefPoint *143*
SetDieResult *133*
SetDieResultAsNum *133*
SetDieStatus *133*
SetExternalMode *224*
SetHeaterSoak *233*
SetHeaterTemp *234*
SetLoaderGate *224*
SetLotID *121*
SetMapHome *121*
SetMapOrientation *121*
SetMapRoute *122*
SetMicroLight *224*
SetOffset *227*
SetOutput *224*
SetProbeHeight *186*
SetProbeHome *187*
SetProbeIndex *187*
SetProbeMode *188*
SetProductID *122*

SetRectMapParams *123*
SetScopeHeight *197*
SetScopeHome *198*
SetScopeIndex *198*
SetScopeMode *199*
SetSoftwareStop *225*
SetStageLock *225*
SetSubDieData *143*
SetSubDieDataAsColRow *144*
SetSubDieDataAsNum *144*
SetSubDieLabel *145*
SetSubDieLabelAsNum *145*
SetSubDieStatus *145*
SetThetaHome *210*
SetTTLLine *101*
SetTypedOutput *226*
SetWaferID *123*
SetWaferMapMode *123*
SetWaferMapParams *123*
SetWaferMapParams2 *124*
SetWaferNum *124*
SetWaferProfileOptions *155*
SetWaferTestAngle *125*
SetWindowState *125*
ShapeTracker *98*
ShowWizard *86*
Simple Sequential Scripting
    SCI commands *52*
    special functions *52*
    using the Command Chooser *52*
SnapImage *98*
SoftwareStopChangedNotify *234*
StartAutomationTemperature *87*
StartReAlign *90*
StartReAlignTemperature *90*
StartScript *81*
StartTTLTest *101*
StartWaferJob *108*
StartWaferProfiling *155*
StepFailedBack *134*
StepFailedClusterBack *150*
StepFailedClusterForward *150*
StepFailedForward *134*
StepFirstCluster *150*
StepFirstDie *134*
StepNextCluster *151*
StepNextDie *135*
StepNextDieOffset *135*
StepNextSubDie *146*
StepToDie *136*
StopAllMovements *226*
StopAutomationTemperature *87*
StopAZoom *79*
StopChuckMovement *174*
StopHeatChuck *234*
StopProbeMovement *189*
StopReAlign *91*
StopReAlignTemperature *91*
StopScopeMovement *199*
StopThetaMovement *211*
StopWaferProfiling *156*
SwitchOffset *87*
SynchronizeCamera *99*
SyncMapHome *125*

T

TransportWafer *108*
TTL pinout *11*
TTL protocol *9*
TTL user interface *11*
TTLTestDone *74*

U

UnloadWafer *109*
UpdateCassetteStatus *109*
UpdateWaferID *109*

V

VueTrackAlign *99*

W

WMNewCurrentDie *74*
WMSetupChange *75*

Corporate Headquarters
7005 Southfront Road
Livermore, CA 94551
Phone: 925-290-4000
www.formfactor.com

FormFactor™