

**WAVEMASTER**  
**WAVEPRO 7000 SERIES**  
**DDA5005**  
**SERIAL DATA ANALYZER**

**AUTOMATION MANUAL**

**JULY 2003**



**LeCroy Corporation**

700 Chestnut Ridge Road  
Chestnut Ridge, NY 10977-6499  
Tel: (845) 578 6020, Fax: (845) 578 5985

**Internet:** [www.lecroy.com](http://www.lecroy.com)

© 2003 by LeCroy Corporation. All rights reserved.

LeCroy, ActiveDSO, ProBus, SMART Trigger, WavePro, Waverunner, and Wave Surfer are registered trademarks of LeCroy Corporation. JitterTrack, WaveMaster, and X-Stream are trademarks of LeCroy Corporation. Information in this publication supersedes all earlier versions. Specifications subject to change without notice.

901783

## CHAPTER 1: ABOUT AUTOMATION

<b>OVERVIEW OF AUTOMATION .....</b>	<b>1-3</b>
Standards .....	1-3
Compatibility with Other LeCroy Scopes .....	1-3
Automation and IEEE 488.2 Remote Control – How Do They Compare? .....	1-3
General Characteristics .....	1-4
<b>INTRODUCTION TO THE X-STREAM BROWSER.....</b>	<b>1-5</b>
<b>STEP-BY-STEP INTRODUCTION TO AUTOMATION USING VBScript.....</b>	<b>1-6</b>
<b>WHERE IS AUTOMATION USED?.....</b>	<b>1-8</b>
<b>SETUPS (PANEL FILES) ARE PROGRAMS!.....</b>	<b>1-9</b>
<b>CUSTOM MATH AND MEASUREMENTS.....</b>	<b>1-11</b>
CustomDSO .....	1-11
Control from External Applications .....	1-12
<b>From Visual Basic.....</b>	<b>1-13</b>
<b>From MATLAB .....</b>	<b>1-13</b>
<b>From MS Office (Excel).....</b>	<b>1-14</b>
<b>CONTROL BY DISTRIBUTED COM (DCOM).....</b>	<b>1-15</b>
Security Settings on the Instrument (Server): User Level .....	1-15
Security Settings on the Instrument (Server): Share Level.....	1-15
Initialize the Controlling PC (Client) .....	1-15
Connecting to a Remote Instrument .....	1-15
<b>CONTROL VARIABLES EXPLAINED.....</b>	<b>1-17</b>
<b>ACCESSING WAVEFORM/MEASUREMENT RESULTS .....</b>	<b>1-19</b>
Waveforms .....	1-19
Measurements .....	1-20
Result Status .....	1-21
<b>SYNCHRONIZATION.....</b>	<b>1-22</b>
<b>GOOD PRACTICES .....</b>	<b>1-24</b>
<b>EXAMPLES.....</b>	<b>1-24</b>
Example 1: Excel Macro to Perform FFT of C1 .....	1-24
Example 2: VBScript Program to Perform FFT of C1 and Store Results in Text File .....	1-25
Example 3: Script to Measure the Rise Time of the Signal on C1 and Display It in a Popup Window .....	1-26
<b>EARLY AND LATE BINDING.....</b>	<b>1-27</b>
<b>VBS REMOTE COMMAND.....</b>	<b>1-28</b>
<b>X-STREAM DSO OBJECTS.....</b>	<b>1-29</b>

## AUTOMATION MANUAL

## CHAPTER 2: CONTROL REFERENCE

<b>LECROY.XSTREAMDSO.1 APP.....</b>	<b>2-2</b>
AddZoomTrace <i>Action</i> .....	2-2
AutoSetup <i>Action</i> .....	2-3
ClearSweeps <i>Action</i> .....	2-3
DoPrint <i>Action</i> .....	2-3
ExitWithoutConfirm <i>Action</i> .....	2-4
FirmwareVersion <i>String</i> .....	2-4
Height <i>Property</i> .....	2-4
HideClock <i>Bool</i> .....	2-4
InstrumentID <i>String</i> .....	2-5
InstrumentModel <i>String</i> .....	2-5
Left <i>Property</i> .....	2-5
Minimize <i>Action</i> .....	2-6
Quit() <i>Method</i> .....	2-6
ResetPreferences <i>Action</i> .....	2-6
SetToDefaultSetup <i>Action</i> .....	2-7
Shutdown <i>Action</i> .....	2-7
Sleep([in] double timeoutMilliseconds) <i>Method</i> .....	2-7
Top <i>Property</i> .....	2-8
TouchScreenEnable <i>Bool</i> .....	2-8
WaitUntilIdle([in] double timeoutSeconds) <i>Method</i> .....	2-8
Width <i>Property</i> .....	2-9
Windowed <i>Action</i> .....	2-9
WindowState <i>Property</i> .....	2-9
<b>ACQUISITION APP.ACQUISITION.....</b>	<b>2-10</b>
Acquire([in] double timeoutSeconds, [in] long) <i>Method</i> .....	2-10
Calibrate <i>Action</i> .....	2-10
ClearSweeps <i>Action</i> .....	2-11
TriggerMode <i>Enum</i> .....	2-11
<b>AUXOUTPUT APP.ACQUISITION.AUXOUTPUT.....</b>	<b>2-13</b>
Amplitude <i>Double</i> .....	2-13
AuxInCoupling <i>Enum</i> .....	2-14
Frequency <i>DoubleLockstep</i> .....	2-14
Mode <i>Enum</i> .....	2-14
PulseWidth <i>Double</i> .....	2-15
SetToTTL <i>Bool</i> .....	2-15
<b>CHANNELS APP.ACQUISITION.CHANNELS.....</b>	<b>2-16</b>
CX <i>app.Acquisition.Cx</i> .....	2-16
AverageSweeps <i>Integer</i> .....	2-17
AxisXRotation <i>Integer</i> .....	2-17
AxisYRotation <i>Integer</i> .....	2-18
BandwidthLimit <i>Enum</i> .....	2-18
ClearSweeps <i>Action</i> .....	2-18
Coupling <i>Enum</i> .....	2-19
Deskew <i>Double</i> .....	2-19
InterpolateType <i>Enum</i> .....	2-19
Invert <i>Bool</i> .....	2-20
LabelsPosition <i>String</i> .....	2-20
LabelsText <i>String</i> .....	2-20
Persist3DQuality <i>Enum</i> .....	2-21
Persisted <i>Bool</i> .....	2-21

## TABLE OF CONTENTS

Persistence3d <i>Bool</i> .....	2-21
PersistenceMonoChrome <i>Bool</i> .....	2-22
PersistenceSaturation <i>Integer</i> .....	2-22
PersistenceTime <i>Enum</i> .....	2-22
ProbeAttenuation <i>Double</i> .....	2-23
ShowLastTrace <i>Bool</i> .....	2-23
UseDotJoin <i>Bool</i> .....	2-23
UseGrid <i>String</i> .....	2-24
VerOffset <i>Double</i> .....	2-24
VerScale <i>DoubleLockstep</i> .....	2-24
VerScaleVariable <i>Bool</i> .....	2-25
View <i>Bool</i> .....	2-25
ViewLabels <i>Bool</i> .....	2-25
<b>RESULTapp.Acquisition.Cx.Out.Result</b> .....	<b>2-26</b>
DataArray <i>Property</i> .....	2-26
FirstEventTime <i>Property</i> .....	2-27
HorizontalFrameStart <i>Property</i> .....	2-27
HorizontalFrameStop <i>Property</i> .....	2-27
HorizontalOffset <i>Property</i> .....	2-28
HorizontalPerStep <i>Property</i> .....	2-28
HorizontalResolution <i>Property</i> .....	2-28
HorizontalUnits <i>Property</i> .....	2-29
IndexOfFirstSampleInFrame <i>Property</i> .....	2-29
LastEventTime <i>Property</i> .....	2-29
NumFrameDimensions <i>Property</i> .....	2-30
NumSamplesInFrame <i>Property</i> .....	2-30
Samples <i>Property</i> .....	2-30
Status <i>Property</i> .....	2-31
StatusDescription <i>Property</i> .....	2-31
Sweeps <i>Property</i> .....	2-31
VerticalFrameStart <i>Property</i> .....	2-31
VerticalFrameStop <i>Property</i> .....	2-32
VerticalMaxPossible <i>Property</i> .....	2-32
VerticalMinPossible <i>Property</i> .....	2-32
VerticalOffset <i>Property</i> .....	2-32
VerticalPerStep <i>Property</i> .....	2-33
VerticalResolution <i>Property</i> .....	2-33
VerticalUnits <i>Property</i> .....	2-33
<b>HORIZONTAL APP.ACQUISITION.HORIZONTAL</b> .....	<b>2-35</b>
AcquisitionDuration <i>Double</i> .....	2-35
ActiveChannels <i>Enum</i> .....	2-36
HorOffset <i>Double</i> .....	2-36
HorOffsetControl <i>Enum</i> .....	2-36
HorOffsetOrigin <i>Double</i> .....	2-37
HorScale <i>DoubleLockstep</i> .....	2-37
HorUnits <i>String</i> .....	2-37
MaxSamples <i>DoubleLockstep</i> .....	2-38
NumPoints <i>Integer</i> .....	2-38
NumSegments <i>Integer</i> .....	2-38
ReferenceClock <i>Enum</i> .....	2-39
SampleClock <i>Enum</i> .....	2-39
SampleMode <i>Enum</i> .....	2-39
SampleRate <i>DoubleLockstep</i> .....	2-40
SamplingRate <i>Double</i> .....	2-40
SequenceTimeout <i>Double</i> .....	2-40

**AUTOMATION MANUAL**

SequenceTimeoutEnable <i>Bool</i> .....	2-40
SmartMemory <i>Enum</i> .....	2-41
TimePerPoint <i>Double</i> .....	2-41
ZeroDelay <i>Action</i> .....	2-41
<b>TRIGGER APP.ACQUISITION.TRIGGER.....</b>	<b>2-42</b>
DropoutTime <i>Double</i> .....	2-42
Glitch <i>Enum</i> .....	2-43
GlitchHigh <i>Double</i> .....	2-43
GlitchLow <i>Double</i> .....	2-43
HoldoffEvents <i>Integer</i> .....	2-44
HoldoffTime <i>Double</i> .....	2-44
HoldoffType <i>Enum</i> .....	2-44
Interval <i>Enum</i> .....	2-45
IntervalDelta <i>Double</i> .....	2-45
IntervalHigh <i>Double</i> .....	2-46
IntervalLow <i>Double</i> .....	2-46
IntervalNominal <i>Double</i> .....	2-46
IntervalRange <i>Enum</i> .....	2-46
PatternType <i>Enum</i> .....	2-46
QualEvents <i>Integer</i> .....	2-47
QualFirst <i>Bool</i> .....	2-47
QualState <i>Enum</i> .....	2-47
QualTime <i>Double</i> .....	2-48
QualWait <i>Enum</i> .....	2-48
Source <i>Enum</i> .....	2-48
TrigLevel <i>Double</i> .....	2-49
Type <i>Enum</i> .....	2-49
ValidateSource <i>Enum</i> .....	2-49
Width <i>Enum</i> .....	2-50
WidthDelta <i>Double</i> .....	2-50
WidthNominal <i>Double</i> .....	2-50
WidthRange <i>Enum</i> .....	2-51
ZeroLevel <i>Action</i> .....	2-51
<b>CX APP.ACQUISITION.TRIGGER.CX.....</b>	<b>2-52</b>
InputImpedance <i>Enum</i> .....	2-52
Level <i>Double</i> .....	2-52
PatternState <i>Enum</i> .....	2-53
Slope <i>Enum</i> .....	2-53
<b>EXT APP.ACQUISITION.TRIGGER.EXT.....</b>	<b>2-54</b>
Coupling <i>Enum</i> .....	2-54
InputImpedance <i>Enum</i> .....	2-54
Level <i>Double</i> .....	2-55
PatternState <i>Enum</i> .....	2-55
Slope <i>Enum</i> .....	2-55
<b>LINE APP.ACQUISITION.TRIGGER.LINE.....</b>	<b>2-56</b>
<b>SOURCES APP.ACQUISITION.TRIGGER.SOURCES.....</b>	<b>2-57</b>
<b>CURSORS APP.CURSORS.....</b>	<b>2-58</b>
Pos1 <i>Double</i> .....	2-58
Pos2 <i>Double</i> .....	2-58
Readout <i>Enum</i> .....	2-59
Track <i>Bool</i> .....	2-59
Type <i>Enum</i> .....	2-59

---

**TABLE OF CONTENTS**


---

View <i>Bool</i> .....	2-60
XPos1 <i>Double</i> .....	2-60
XPos2 <i>Double</i> .....	2-60
YPos1 <i>Double</i> .....	2-60
YPos2 <i>Double</i> .....	2-61
<b>CUSTOMDSO APP.CUSTOMDSO.....</b>	<b>2-62</b>
ActionEnable1 <i>Bool</i> .....	2-62
ActionEnable2 <i>Bool</i> .....	2-62
ActionEnable3 <i>Bool</i> .....	2-63
ActionEnable4 <i>Bool</i> .....	2-63
ActionEnable5 <i>Bool</i> .....	2-63
ActionEnable6 <i>Bool</i> .....	2-63
ActionEnable7 <i>Bool</i> .....	2-63
ActionEnable8 <i>Bool</i> .....	2-63
ActionScript1 <i>FileName</i> .....	2-63
ActionScript2 <i>FileName</i> .....	2-63
ActionScript3 <i>FileName</i> .....	2-64
ActionScript4 <i>FileName</i> .....	2-64
ActionScript5 <i>FileName</i> .....	2-64
ActionScript6 <i>FileName</i> .....	2-64
ActionScript7 <i>FileName</i> .....	2-64
ActionScript8 <i>FileName</i> .....	2-64
Mode <i>Enum</i> .....	2-64
PlugIn1Install <i>Action</i> .....	2-65
PlugIn1ProgId <i>String</i> .....	2-65
PlugIn1Remove <i>Action</i> .....	2-65
PresentAtPowerUp <i>Bool</i> .....	2-65
<b>DISPLAY APP.DISPLAY.....</b>	<b>2-66</b>
AxisLabels <i>Bool</i> .....	2-67
AxisXRotation <i>Integer</i> .....	2-67
AxisYRotation <i>Integer</i> .....	2-67
C1Color <i>Color</i> .....	2-68
C1PrintColor <i>Color</i> .....	2-68
C2Color <i>Color</i> .....	2-69
C2PrintColor <i>Color</i> .....	2-69
C3Color <i>Color</i> .....	2-69
C3PrintColor <i>Color</i> .....	2-69
C4Color <i>Color</i> .....	2-69
C4PrintColor <i>Color</i> .....	2-69
ClearSweeps <i>Action</i> .....	2-69
DisplayMode <i>Enum</i> .....	2-70
F1Color <i>Color</i> .....	2-70
F1PrintColor <i>Color</i> .....	2-70
F2Color <i>Color</i> .....	2-70
F2PrintColor <i>Color</i> .....	2-70
F3Color <i>Color</i> .....	2-70
F3PrintColor <i>Color</i> .....	2-70
F4Color <i>Color</i> .....	2-71
F4PrintColor <i>Color</i> .....	2-71
F5Color <i>Color</i> .....	2-71
F5PrintColor <i>Color</i> .....	2-71
F6Color <i>Color</i> .....	2-71
F6PrintColor <i>Color</i> .....	2-71
F7Color <i>Color</i> .....	2-71

**AUTOMATION MANUAL**

F7PrintColor <i>Color</i> .....	2-71
F8Color <i>Color</i> .....	2-71
F8PrintColor <i>Color</i> .....	2-72
FactoryDefault <i>Action</i> .....	2-72
GridIntensity <i>Integer</i> .....	2-72
GridMode <i>Enum</i> .....	2-72
GridOnTop <i>Bool</i> .....	2-73
LockPersistence <i>Enum</i> .....	2-73
M1Color <i>Color</i> .....	2-73
M1PrintColor <i>Color</i> .....	2-73
M2Color <i>Color</i> .....	2-73
M2PrintColor <i>Color</i> .....	2-74
M3Color <i>Color</i> .....	2-74
M3PrintColor <i>Color</i> .....	2-74
M4Color <i>Color</i> .....	2-74
M4PrintColor <i>Color</i> .....	2-74
NumSegmentsDisplayed <i>Integer</i> .....	2-74
Persist3DQuality <i>Enum</i> .....	2-74
Persisted <i>Bool</i> .....	2-75
Persistence3d <i>Bool</i> .....	2-75
PersistenceLastTrace <i>Bool</i> .....	2-75
PersistenceMonoChrome <i>Bool</i> .....	2-76
PersistenceSaturation <i>Integer</i> .....	2-76
PersistenceStyle <i>Enum</i> .....	2-76
PersistenceTime <i>Enum</i> .....	2-76
PreviewPrintColors <i>Action</i> .....	2-77
ResetAll <i>Action</i> .....	2-77
SegmentMode <i>Enum</i> .....	2-77
StartSegment <i>Integer</i> .....	2-78
TraceStyle <i>Enum</i> .....	2-78
<b>HARDCOPY APP.HARDCOPY.....</b>	<b>2-79</b>
Destination <i>Enum</i> .....	2-79
Directory <i>FileName</i> .....	2-79
EMailMessage <i>String</i> .....	2-80
GridAreaOnly <i>Bool</i> .....	2-80
HardcopyArea <i>Enum</i> .....	2-80
ImageFileFormat <i>Enum</i> .....	2-81
Orientation <i>Enum</i> .....	2-81
PreferredFilenameString.....	2-81
Print <i>Action</i> .....	2-82
PromptForMessage <i>Bool</i> .....	2-82
SelectedPrinter <i>Enum</i> .....	2-82
StripChart <i>Bool</i> .....	2-82
StripChartFactor <i>Enum</i> .....	2-82
UsePrintPalette <i>Bool</i> .....	2-83
<b>MATH APP.MATH.....</b>	<b>2-84</b>
ClearSweeps <i>Action</i> .....	2-84
ResetZoom <i>Action</i> .....	2-84
<b>FUNCTIONS APP.MATH.FUNCTIONS.....</b>	<b>2-85</b>
FX <i>app.Math.Fx</i> .....	2-85
AxisXRotation <i>Integer</i> .....	2-85
AxisYRotation <i>Integer</i> .....	2-86
ClearSweeps <i>Action</i> .....	2-86
DoResetZoom <i>Action</i> .....	2-86

## TABLE OF CONTENTS

DoStoreToMemoryTrace <i>Action</i> .....	2-86
Equation <i>String</i> .....	2-87
GraphOp <i>Enum</i> .....	2-87
LabelsPosition <i>String</i> .....	2-87
LabelsText <i>String</i> .....	2-88
MathMode <i>Enum</i> .....	2-88
MeasureOp <i>Enum</i> .....	2-88
Operator1 <i>Enum</i> .....	2-90
Operator2 <i>Enum</i> .....	2-92
Persist3DQuality <i>Enum</i> .....	2-93
Persisted <i>Bool</i> .....	2-93
Persistence3d <i>Bool</i> .....	2-94
PersistenceMonoChrome <i>Bool</i> .....	2-94
PersistenceSaturation <i>Integer</i> .....	2-94
PersistenceTime <i>Enum</i> .....	2-94
ShowLastTrace <i>Bool</i> .....	2-95
Source1 <i>Enum</i> .....	2-95
Source2 <i>Enum</i> .....	2-96
Source3 <i>Enum</i> .....	2-97
UseDotJoin <i>Bool</i> .....	2-97
UseGrid <i>String</i> .....	2-98
View <i>Bool</i> .....	2-98
ViewLabels <i>Bool</i> .....	2-98
<b>OPERATOR1SETUP APP.MATH.FX.OPERATOR1SETUP.....</b>	<b>2-99</b>
<b>RESULT app.Math.Fx.Out.Result.....</b>	<b>2-100</b>
<b>ZOOM APP.MATH.FX.ZOOM.....</b>	<b>2-101</b>
HorPos <i>Double</i> .....	2-101
HorZoom <i>Double</i> .....	2-101
ResetZoom <i>Action</i> .....	2-101
VariableHorZoom <i>Bool</i> .....	2-102
VariableVerZoom <i>Bool</i> .....	2-102
VerPos <i>Double</i> .....	2-102
VerZoom <i>Double</i> .....	2-102
<b>XY APP.MATH.XY.....</b>	<b>2-103</b>
AxisXRotation <i>Integer</i> .....	2-103
AxisYRotation <i>Integer</i> .....	2-103
ClearSweeps <i>Action</i> .....	2-104
InputX <i>Enum</i> .....	2-104
InputY <i>Enum</i> .....	2-104
Persist3DQuality <i>Enum</i> .....	2-105
Persisted <i>Bool</i> .....	2-105
Persistence3d <i>Bool</i> .....	2-105
PersistenceMonoChrome <i>Bool</i> .....	2-105
PersistenceSaturation <i>Integer</i> .....	2-105
PersistenceTime <i>Enum</i> .....	2-105
ShowLastTrace <i>Bool</i> .....	2-106
UseDotJoin <i>Bool</i> .....	2-106
<b>RESULT app.Math.XY.Out.Result.....</b>	<b>2-107</b>
DataArray <i>Property</i> .....	2-107
FirstEventTime <i>Property</i> .....	2-107
HorizontalFrameStart <i>Property</i> .....	2-108
HorizontalFrameStop <i>Property</i> .....	2-108
HorizontalOffset <i>Property</i> .....	2-108
HorizontalPerStep <i>Property</i> .....	2-108

**AUTOMATION MANUAL**

HorizontalResolution <i>Property</i> .....	2-109
HorizontalUnits <i>Property</i> .....	2-109
LastEventTime <i>Property</i> .....	2-109
NumFrameDimensions <i>Property</i> .....	2-110
Samples <i>Property</i> .....	2-110
Sweeps <i>Property</i> .....	2-110
XFrameStart <i>Property</i> .....	2-110
XFrameStop <i>Property</i> .....	2-111
XMaxPossible <i>Property</i> .....	2-111
XMinPossible <i>Property</i> .....	2-111
XOffset <i>Property</i> .....	2-111
XPerStep <i>Property</i> .....	2-111
XResolution <i>Property</i> .....	2-112
XUnits <i>Property</i> .....	2-112
YFrameStart <i>Property</i> .....	2-112
YFrameStop <i>Property</i> .....	2-113
YMaxPossible <i>Property</i> .....	2-113
YMinPossible <i>Property</i> .....	2-113
YOffset <i>Property</i> .....	2-113
YPerStep <i>Property</i> .....	2-113
YResolution <i>Property</i> .....	2-114
YUnits <i>Property</i> .....	2-114
<b>MEASURE APP.MEASURE.....</b>	<b>2-115</b>
ClearAll <i>Action</i> .....	2-115
ClearSweeps <i>Action</i> .....	2-115
HelpMarkers <i>Enum</i> .....	2-116
HistoOn <i>Bool</i> .....	2-116
MeasureMode <i>Enum</i> .....	2-116
SetGateToDefault <i>Action</i> .....	2-116
StatsOn <i>Bool</i> .....	2-116
StdGateStart <i>Double</i> .....	2-116
StdGateStop <i>Double</i> .....	2-117
StdSource <i>Enum</i> .....	2-117
<b>MEASURE APP.MEASURE.MEASURE.....</b>	<b>2-118</b>
PX <i>app.Measure.Px</i> .....	2-118
ArithEngine <i>Enum</i> .....	2-118
Equation <i>String</i> .....	2-119
GateStart <i>Double</i> .....	2-119
GateStop <i>Double</i> .....	2-119
MeasurementType <i>Enum</i> .....	2-119
ParamEngine <i>Enum</i> .....	2-120
PSource1 <i>Enum</i> .....	2-121
PSource2 <i>Enum</i> .....	2-122
Source1 <i>Enum</i> .....	2-122
Source2 <i>Enum</i> .....	2-122
View <i>Bool</i> .....	2-123
<b>OPERATOR APP.MEASURE.PX.OPERATOR.....</b>	<b>2-124</b>
<b>RESULT APP.MEASURE.PX.OUT.RESULT.....</b>	<b>2-125</b>
HorizontalResolution <i>Property</i> .....	2-125
HorizontalUnits <i>Property</i> .....	2-125
Value <i>Property</i> .....	2-125
VerticalResolution <i>Property</i> .....	2-126
VerticalUnits <i>Property</i> .....	2-126

## TABLE OF CONTENTS

<b>STATISTICS APP.MEASURE.PX.STATISTICS.....</b>	<b>2-127</b>
RESULT <i>app.Measure.Px.Statistics("histo").Result</i> .....	2-127
Base <i>Property</i> .....	2-127
BinPopulations <i>Property</i> .....	2-128
Bins <i>Property</i> .....	2-128
BinWidth <i>Property</i> .....	2-128
FirstPopulatedBin <i>Property</i> .....	2-129
HorizontalFrameStart <i>Property</i> .....	2-129
HorizontalFrameStop <i>Property</i> .....	2-129
HorizontalUnits <i>Property</i> .....	2-129
LastPopulatedBin <i>Property</i> .....	2-130
Max <i>Property</i> .....	2-130
MaxPopulation <i>Property</i> .....	2-130
MaxPopulationBin <i>Property</i> .....	2-130
Mean <i>Property</i> .....	2-131
Min <i>Property</i> .....	2-131
OffsetAtLeftEdge <i>Property</i> .....	2-131
PeakInfo <i>Property</i> .....	2-131
Peaks <i>Property</i> .....	2-132
PopulationInside <i>Property</i> .....	2-132
PopulationOver <i>Property</i> .....	2-132
PopulationUnder <i>Property</i> .....	2-133
Rms <i>Property</i> .....	2-133
Sdev <i>Property</i> .....	2-133
Sweeps <i>Property</i> .....	2-133
Top <i>Property</i> .....	2-134
VerticalFrameStart <i>Property</i> .....	2-134
VerticalFrameStop <i>Property</i> .....	2-134
VerticalUnits <i>Property</i> .....	2-134
<b>RESULT <i>app.Measure.Px.Statistics("last").Result</i>.....</b>	<b>2-135</b>
Value <i>Property</i> .....	2-135
ValueArray <i>Property</i> .....	2-135
VerticalResolution <i>Property</i> .....	2-135
VerticalUnits <i>Property</i> .....	2-135
<b>RESULT <i>app.Measure.Px.Statistics("max").Result</i>.....</b>	<b>2-136</b>
HorizontalResolution <i>Property</i> .....	2-136
HorizontalUnits <i>Property</i> .....	2-136
Value <i>Property</i> .....	2-136
VerticalResolution <i>Property</i> .....	2-137
VerticalUnits <i>Property</i> .....	2-137
<b>RESULT APP.MEASURE.PX.STATISTICS("MEAN").RESULT .....</b>	<b>2-138</b>
HorizontalResolution <i>Property</i> .....	2-138
HorizontalUnits <i>Property</i> .....	2-138
Value <i>Property</i> .....	2-138
VerticalResolution <i>Property</i> .....	2-139
VerticalUnits <i>Property</i> .....	2-139
<b>RESULT <i>app.Measure.Px.Statistics("min").Result</i>.....</b>	<b>2-140</b>
HorizontalResolution <i>Property</i> .....	2-140
HorizontalUnits <i>Property</i> .....	2-140
Value <i>Property</i> .....	2-140
VerticalResolution <i>Property</i> .....	2-141
VerticalUnits <i>Property</i> .....	2-141
<b>RESULT <i>app.Measure.Px.Statistics("num").Result</i>.....</b>	<b>2-142</b>
HorizontalResolution <i>Property</i> .....	2-142
HorizontalUnits <i>Property</i> .....	2-142

**AUTOMATION MANUAL**

Value Property.....	2-142
VerticalResolution Property.....	2-143
VerticalUnits Property.....	2-143
<b>RESULT app.Measure.Px.Statistics("sdev").Result.....</b>	<b>2-144</b>
HorizontalResolution Property.....	2-144
HorizontalUnits Property.....	2-144
Value Property.....	2-144
VerticalResolution Property.....	2-145
VerticalUnits Property.....	2-145
<b>MEMORY APP.MEMORY.....</b>	<b>2-146</b>
ClearAllMem Action.....	2-146
<b>MEMORIES APP.MEMORY.MEMORIES.....</b>	<b>2-147</b>
<b>MX APP.MEMORY.MX.....</b>	<b>2-148</b>
ClearMem Action.....	2-148
Copy Action.....	2-148
IncludedInMZgroup Bool.....	2-148
LabelsPosition String.....	2-148
LabelsText String.....	2-148
Source1 Enum.....	2-148
SourceTimeText String.....	2-149
UseDotJoin Bool.....	2-149
UseGrid String.....	2-149
UserText String.....	2-149
View Bool.....	2-149
ViewLabels Bool.....	2-150
RESULT app.Memory.Mx.Out.Result.....	2-150
<b>ZOOM APP.MEMORY.MX.ZOOM.....</b>	<b>2-151</b>
HorPos Double.....	2-151
HorZoom Double.....	2-151
ResetZoom Action.....	2-152
VariableHorZoom Bool.....	2-152
VariableVerZoom Bool.....	2-152
VerPos Double.....	2-152
VerZoom Double.....	2-152
<b>PASSFAIL APP.PASSFAIL.....</b>	<b>2-153</b>
ActionOn Enum.....	2-154
Alarm Bool.....	2-154
EnableActions Bool.....	2-154
ExecuteScript Bool.....	2-154
PredefinedConditions Enum.....	2-154
PrintScreen Bool.....	2-154
Pulse Bool.....	2-154
Save Bool.....	2-155
Stop Bool.....	2-155
StopAfter Integer.....	2-155
StopTesting Bool.....	2-155
Testing Bool.....	2-155
<b>QX APP.PASSFAIL.QX.....</b>	<b>2-156</b>
ConditionEngine Enum.....	2-156
Equation String.....	2-156
Source1 Enum.....	2-156
Source2 Enum.....	2-156

---

**T A B L E O F C O N T E N T S**


---

View <i>Bool</i> .....	2-157
<b>RESULT <i>app.PassFail.Qx.Out.Result</i>.....</b>	<b>2-158</b>
HorizontalResolution <i>Property</i> .....	2-158
HorizontalUnits <i>Property</i> .....	2-158
Status <i>Property</i> .....	2-158
StatusDescription <i>Property</i> .....	2-158
Value <i>Property</i> .....	2-158
ValueArray <i>Property</i> .....	2-158
<b>PREFERENCES <i>APP.PREFERENCES</i>.....</b>	<b>2-159</b>
AudibleFeedback <i>Bool</i> .....	2-159
OffsetControl <i>Enum</i> .....	2-159
Performance <i>Enum</i> .....	2-159
<b>EMAIL <i>APP.PREFERENCES.EMAIL</i>.....</b>	<b>2-160</b>
DefaultRecipient <i>String</i> .....	2-160
Mode <i>Enum</i> .....	2-160
OriginatorAddress <i>String</i> .....	2-160
SendTestMail <i>Action</i> .....	2-160
SMTPServer <i>String</i> .....	2-160
<b>SAVERECALL <i>APP.SAVERECALL</i>.....</b>	<b>2-162</b>
<b>SETUP <i>APP.SAVERECALL.SETUP</i>.....</b>	<b>2-163</b>
DoRecallDefaultNvlPanel <i>Action</i> .....	2-163
DoRecallDefaultPanel <i>Action</i> .....	2-164
DoRecallPanel <i>Action</i> .....	2-164
DoSavePanel <i>Action</i> .....	2-164
InternalName1 <i>String</i> .....	2-165
InternalName2 <i>String</i> .....	2-165
InternalName3 <i>String</i> .....	2-165
InternalName4 <i>String</i> .....	2-165
InternalName5 <i>String</i> .....	2-165
InternalName6 <i>String</i> .....	2-165
PanelFilename <i>FileName</i> .....	2-165
RecallInternal1 <i>Action</i> .....	2-165
RecallInternal2 <i>Action</i> .....	2-166
RecallInternal3 <i>Action</i> .....	2-166
RecallInternal4 <i>Action</i> .....	2-166
RecallInternal5 <i>Action</i> .....	2-166
RecallInternal6 <i>Action</i> .....	2-166
SaveInternal1 <i>Action</i> .....	2-166
SaveInternal2 <i>Action</i> .....	2-166
SaveInternal3 <i>Action</i> .....	2-167
SaveInternal4 <i>Action</i> .....	2-167
SaveInternal5 <i>Action</i> .....	2-167
SaveInternal6 <i>Action</i> .....	2-167
<b>UTILITIES <i>APP.SAVERECALL.UTILITIES</i>.....</b>	<b>2-168</b>
CreateDir <i>Action</i> .....	2-168
DeleteAll <i>Action</i> .....	2-168
DeleteAllWithPrompt <i>Action</i> .....	2-168
DeleteFile <i>Action</i> .....	2-169
<b>WAVEFORM <i>APP.SAVERECALL.WAVEFORM</i>.....</b>	<b>2-170</b>
Delimiter <i>Enum</i> .....	2-171
DoRecall <i>Action</i> .....	2-171
DoSave <i>Action</i> .....	2-171

**AUTOMATION MANUAL**

RecallDestination <i>Enum</i> .....	2-171
RecallFrom <i>Enum</i> .....	2-171
RecallSource <i>Enum</i> .....	2-171
SaveDestination <i>Enum</i> .....	2-172
SaveSource <i>Enum</i> .....	2-172
SaveTo <i>Enum</i> .....	2-173
SubFormat <i>Enum</i> .....	2-173
TraceTitle <i>String</i> .....	2-173
WaveFormat <i>Enum</i> .....	2-174
WaveformDir <i>FileName</i> .....	2-174
<b>SDA APP.SDA.....</b>	<b>2-175</b>
BERParamsOn <i>Bool</i> .....	2-175
CalcType <i>Enum</i> .....	2-175
ClockSource <i>Enum</i> .....	2-176
DarkCalLevel <i>Double</i> .....	2-176
ErrorMapOn <i>Bool</i> .....	2-177
EyeMode <i>Enum</i> .....	2-178
FindFrequency <i>Action</i> .....	2-178
FindScale <i>Action</i> .....	2-178
HiPassFreq <i>Double</i> .....	2-178
LowPassFreq <i>Double</i> .....	2-178
MaskFailTraceOn <i>Bool</i> .....	2-178
MaskFileName <i>String</i> .....	2-179
NumPatternBits <i>Integer</i> .....	2-179
PLLOn <i>Bool</i> .....	2-179
RefReceiver <i>Bool</i> .....	2-180
SDAMode <i>Enum</i> .....	2-180
ShowFailLocation <i>Bool</i> .....	2-180
SignalFrequency <i>Double</i> .....	2-181
SignalMode <i>Enum</i> .....	2-181
SignalType <i>Enum</i> .....	2-181
StartN <i>Integer</i> .....	2-182
StepN <i>Integer</i> .....	2-182
StopN <i>Integer</i> .....	2-182
Units <i>Enum</i> .....	2-183
UserSignal <i>Enum</i> .....	2-183
VerAutoFit <i>Bool</i> .....	2-184
<b>BADBITS APP.SDA.BADBITS.....</b>	<b>2-185</b>
AbsLevel <i>Double</i> .....	2-186
AxisXRotation <i>Integer</i> .....	2-186
AxisYRotation <i>Integer</i> .....	2-186
BitsInLocator <i>Integer</i> .....	2-186
C1ReceiverStandard <i>Enum</i> .....	2-186
C2ReceiverStandard <i>Enum</i> .....	2-186
C3ReceiverStandard <i>Enum</i> .....	2-187
C4ReceiverStandard <i>Enum</i> .....	2-187
ClearSweeps <i>Action</i> .....	2-187
ClockTIESlope <i>Enum</i> .....	2-187
DarkCal <i>Action</i> .....	2-188
DarkCalLevel <i>Double</i> .....	2-188
DataSource <i>Enum</i> .....	2-188
EyeMode <i>Enum</i> .....	2-188
EyeThresholdType <i>Enum</i> .....	2-188
FailCursorsOn <i>Bool</i> .....	2-189

## TABLE OF CONTENTS

FailedList <i>Enum</i> .....	2-189
FailedSymbolsFilter <i>Enum</i> .....	2-189
LabelsPosition <i>String</i> .....	2-189
LabelText <i>String</i> .....	2-189
MaskFailTraceOn <i>Bool</i> .....	2-189
MaskFailX <i>Double</i> .....	2-189
MaskFailY <i>Double</i> .....	2-189
MaskType <i>Enum</i> .....	2-189
MaxFailures <i>Integer</i> .....	2-190
MeasurementMode <i>Enum</i> .....	2-190
MonochromeEye <i>Enum</i> .....	2-190
PercentLevel <i>Integer</i> .....	2-190
Persist3DQuality <i>Enum</i> .....	2-190
Persisted <i>Bool</i> .....	2-190
Persistence3d <i>Bool</i> .....	2-190
PersistenceMonoChrome <i>Bool</i> .....	2-190
PersistenceSaturation <i>Integer</i> .....	2-190
PersistenceTime <i>Enum</i> .....	2-191
RefReceiver <i>Bool</i> .....	2-191
SDAMode <i>Enum</i> .....	2-191
ShowFailLocation <i>Bool</i> .....	2-191
ShowLastTrace <i>Bool</i> .....	2-191
SignalFrequency <i>Double</i> .....	2-191
SignalType <i>Enum</i> .....	2-191
SliceWidth <i>Integer</i> .....	2-192
Stop <i>Bool</i> .....	2-192
TrackMaskFail <i>Action</i> .....	2-192
UseDotJoin <i>Bool</i> .....	2-192
UseGrid <i>String</i> .....	2-193
UserSignal <i>Enum</i> .....	2-193
VerAutoFit <i>Bool</i> .....	2-193
View <i>Bool</i> .....	2-193
ViewLabels <i>Bool</i> .....	2-193
XMargin <i>Integer</i> .....	2-193
YMargin <i>Integer</i> .....	2-194
<b>RESULT app.SDA.BadBits.Out.Result</b> .....	<b>2-195</b>
DataArray <i>Property</i> .....	2-195
ExtendedStatus <i>Property</i> .....	2-195
FirstEventTime <i>Property</i> .....	2-195
HorizontalFrameStart <i>Property</i> .....	2-195
HorizontalFrameStop <i>Property</i> .....	2-196
HorizontalOffset <i>Property</i> .....	2-196
HorizontalPerStep <i>Property</i> .....	2-196
HorizontalResolution <i>Property</i> .....	2-196
HorizontalUnits <i>Property</i> .....	2-196
HorizontalVarianceArray <i>Property</i> .....	2-196
HorizontalVariances <i>Property</i> .....	2-196
IndexOfFirstSampleInFrame <i>Property</i> .....	2-196
LastEventTime <i>Property</i> .....	2-196
NumFrameDimensions <i>Property</i> .....	2-196
NumSamplesInFrame <i>Property</i> .....	2-196
Samples <i>Property</i> .....	2-197
Status <i>Property</i> .....	2-197
StatusDescription <i>Property</i> .....	2-197
Sweeps <i>Property</i> .....	2-197
UpdateTime <i>Property</i> .....	2-197

## AUTOMATION MANUAL

---

VerticalFrameStart <i>Property</i> .....	2-197
VerticalFrameStop <i>Property</i> .....	2-197
VerticalMaxPossible <i>Property</i> .....	2-197
VerticalMinPossible <i>Property</i> .....	2-197
VerticalOffset <i>Property</i> .....	2-197
VerticalPerStep <i>Property</i> .....	2-197
VerticalResolution <i>Property</i> .....	2-198
VerticalUnits <i>Property</i> .....	2-198
<b>BITS APP.SDA.BITS.....</b>	<b>2-199</b>
AxisXRotation <i>Integer</i> .....	2-199
AxisYRotation <i>Integer</i> .....	2-199
ClearSweeps <i>Action</i> .....	2-199
EyeMode <i>Enum</i> .....	2-199
LabelsPosition <i>String</i> .....	2-199
LabelsText <i>String</i> .....	2-200
MaskType <i>Enum</i> .....	2-200
Persist3DQuality <i>Enum</i> .....	2-200
Persisted <i>Bool</i> .....	2-200
Persistence3d <i>Bool</i> .....	2-200
PersistenceMonoChrome <i>Bool</i> .....	2-200
PersistenceSaturation <i>Integer</i> .....	2-200
PersistenceTime <i>Enum</i> .....	2-200
ShowLastTrace <i>Bool</i> .....	2-201
SignalFrequency <i>Double</i> .....	2-201
SignalType <i>Enum</i> .....	2-201
UseDotJoin <i>Bool</i> .....	2-202
UseGrid <i>String</i> .....	2-202
VerAutoFit <i>Bool</i> .....	2-202
View <i>Bool</i> .....	2-202
ViewLabels <i>Bool</i> .....	2-202
<b>RESULT app.SDA.Bits.Out.Result.....</b>	<b>2-203</b>
DataArray <i>Property</i> .....	2-203
ExtendedStatus <i>Property</i> .....	2-203
FirstEventTime <i>Property</i> .....	2-203
HorizontalFrameStart <i>Property</i> .....	2-204
HorizontalFrameStop <i>Property</i> .....	2-204
HorizontalOffset <i>Property</i> .....	2-204
HorizontalPerStep <i>Property</i> .....	2-204
HorizontalResolution <i>Property</i> .....	2-204
HorizontalUnits <i>Property</i> .....	2-204
HorizontalVarianceArray <i>Property</i> .....	2-204
HorizontalVariances <i>Property</i> .....	2-204
IndexOfFirstSampleInFrame <i>Property</i> .....	2-204
LastEventTime <i>Property</i> .....	2-204
NumFrameDimensions <i>Property</i> .....	2-204
NumSamplesInFrame <i>Property</i> .....	2-205
Samples <i>Property</i> .....	2-205
Status <i>Property</i> .....	2-205
StatusDescription <i>Property</i> .....	2-205
Sweeps <i>Property</i> .....	2-205
UpdateTime <i>Property</i> .....	2-205
VerticalFrameStart <i>Property</i> .....	2-205
VerticalFrameStop <i>Property</i> .....	2-205
VerticalMaxPossible <i>Property</i> .....	2-205
VerticalMinPossible <i>Property</i> .....	2-205

## TABLE OF CONTENTS

VerticalOffset <i>Property</i> .....	2-205
VerticalPerStep <i>Property</i> .....	2-206
VerticalResolution <i>Property</i> .....	2-206
VerticalUnits <i>Property</i> .....	2-206
<b>EYE APP.SDA.EYE.....</b>	<b>2-207</b>
AxisXRotation <i>Integer</i> .....	2-207
AxisYRotation <i>Integer</i> .....	2-207
ClearSweeps <i>Action</i> .....	2-207
DataSource <i>Enum</i> .....	2-207
EyeMode <i>Enum</i> .....	2-208
LabelsPosition <i>String</i> .....	2-208
LabelsText <i>String</i> .....	2-208
MaskType <i>Enum</i> .....	2-208
Persist3DQuality <i>Enum</i> .....	2-209
Persisted <i>Bool</i> .....	2-209
Persistence3d <i>Bool</i> .....	2-209
PersistenceMonoChrome <i>Bool</i> .....	2-209
PersistenceSaturation <i>Integer</i> .....	2-209
PersistenceTime <i>Enum</i> .....	2-209
ShowLastTrace <i>Bool</i> .....	2-209
SignalFrequency <i>Double</i> .....	2-209
SignalType <i>Enum</i> .....	2-210
UseDotJoin <i>Bool</i> .....	2-210
UseGrid <i>String</i> .....	2-210
UseMonochrome <i>Bool</i> .....	2-210
VerAutoFit <i>Bool</i> .....	2-211
View <i>Bool</i> .....	2-211
ViewLabels <i>Bool</i> .....	2-211
<b>RESULT app.SDA.Eye.Out.Result.....</b>	<b>2-212</b>
Columns <i>Property</i> .....	2-212
DataArray <i>Property</i> .....	2-212
FirstEventTime <i>Property</i> .....	2-212
HorizontalFrameStart <i>Property</i> .....	2-213
HorizontalFrameStop <i>Property</i> .....	2-213
HorizontalOffset <i>Property</i> .....	2-213
HorizontalPerColumn <i>Property</i> .....	2-213
HorizontalUnits <i>Property</i> .....	2-213
LastEventTime <i>Property</i> .....	2-213
MaxPopulationInRectangle <i>Property</i> .....	2-213
NumFrameDimensions <i>Property</i> .....	2-214
PopulationOfRectangle <i>Property</i> .....	2-214
Rows <i>Property</i> .....	2-214
Sweeps <i>Property</i> .....	2-214
UpdateTime <i>Property</i> .....	2-214
VerticalFrameStart <i>Property</i> .....	2-214
VerticalFrameStop <i>Property</i> .....	2-214
VerticalOffset <i>Property</i> .....	2-215
VerticalPerRow <i>Property</i> .....	2-215
VerticalUnits <i>Property</i> .....	2-215
<b>TIE APP.SDA.TIE.....</b>	<b>2-216</b>
AbsLevel <i>Double</i> .....	2-216
EyeThresholdType <i>Enum</i> .....	2-216
FindFrequency <i>Action</i> .....	2-216
PercentLevel <i>Integer</i> .....	2-216

## AUTOMATION MANUAL

---

PLLOn <i>Bool</i> .....	2-217
SignalFrequency <i>Double</i> .....	2-217
Slope <i>Enum</i> .....	2-217
View <i>Bool</i> .....	2-217
<b>RESULT</b> <i>app.SDA.TIE.Out.Result</i> .....	<b>2-218</b>
ExtendedStatus <i>Property</i> .....	2-218
FirstEventTime <i>Property</i> .....	2-218
HorizontalResolution <i>Property</i> .....	2-218
HorizontalUnits <i>Property</i> .....	2-218
LastEventTime <i>Property</i> .....	2-218
NumFrameDimensions <i>Property</i> .....	2-218
Status <i>Property</i> .....	2-218
StatusDescription <i>Property</i> .....	2-219
UpdateTime <i>Property</i> .....	2-219
Value <i>Property</i> .....	2-219
ValueArray <i>Property</i> .....	2-219
VerticalResolution <i>Property</i> .....	2-219
VerticalUnits <i>Property</i> .....	2-219
<b>DATETIMESETUP</b> <i>APP.UTILITY.DATETIMESETUP</i> .....	<b>2-220</b>
CurrentDateAndTime <i>String</i> .....	2-220
Day <i>Integer</i> .....	2-220
Hour <i>Integer</i> .....	2-221
Minute <i>Integer</i> .....	2-221
Month <i>Integer</i> .....	2-221
Second <i>Integer</i> .....	2-222
SetFromSNTP <i>Action</i> .....	2-222
Validate <i>Action</i> .....	2-222
Year <i>Integer</i> .....	2-223
<b>OPTIONS</b> <i>APP.UTILITY.OPTIONS</i> .....	<b>2-224</b>
InstalledHWOPTIONS <i>String</i> .....	2-224
InstalledSWOptions <i>String</i> .....	2-224
ScopeID <i>String</i> .....	2-224
<b>REMOTE</b> <i>APP.UTILITY.REMOTE</i> .....	<b>2-225</b>
AllowControlFrom <i>String</i> .....	2-225
Assistant <i>Enum</i> .....	2-225
GpibAddress <i>Integer</i> .....	2-225
Interface <i>Enum</i> .....	2-225
RestrictControl <i>Enum</i> .....	2-225
SetToErrorsOnlyAndClearAtStartup <i>Bool</i> .....	2-226
<b>WEBEDITOR</b> <i>APP.WEBEDITOR</i> .....	<b>2-227</b>
AddConnection([in] VARIANT destProcessor, [in] VARIANT Method destInputPin, [in] VARIANT sourceProcessor, [in] VARIANT sourceOutputPin).....	2-228
AddPreview([in] VARIANT sourceProcessor, [in] VARIANT Method sourcePin, [in] BSTR previewName, [in] double xPosition, [in] double xPosition).....	2-228
AddProcessor([in] VARIANT processorOrClassId, [in] BSTR Method requestedName, [in] double xPosition, [in] double yPosition).....	2-228
ClearSweeps <i>Action</i> .....	2-228
GetProcessor([in] VARIANT processor) <i>Method</i> .....	2-228
RemoveAll() <i>Method</i> .....	2-229
RemoveConnection([in] VARIANT destProcessor, [in] VARIANT destInputPin) <i>Method</i> .....	2-229
RemovePreview([in] VARIANT processor) <i>Method</i> .....	2-229
RemoveProcessor([in] VARIANT processor) <i>Method</i> .....	2-229

## CHAPTER 3: MATH/MEASURE CONTROL REFERENCE

<b>AVERAGE</b> <i>APP.MATH.FX.OPERATORYSETUP (OPERATOR = "AVERAGE")</i> .....	<b>3-2</b>
<b>AVERAGE</b> <i>APP.MATH.FX.OPERATORYSETUP (OPERATOR = "AVERAGE")</i> .....	<b>3-3</b>
AverageType <i>Enum</i> .....	3-3
ClearSweeps <i>Action</i> .....	3-3
Sweeps <i>Integer</i> .....	34
<b>BOXCAR</b> <i>APP.MATH.FX.OPERATORYSETUP (OPERATOR = "BOXCAR")</i> .....	<b>3-5</b>
Length <i>Integer</i> .....	3-5
<b>CORRELATION</b> <i>APP.MATH.FX.OPERATORYSETUP (OPERATOR = "CORRELATION")</i> .....	<b>3-6</b>
CorrLength <i>Double</i> .....	3-6
CorrStart <i>Double</i> .....	3-6
<b>DERIVATIVE</b> <i>APP.MATH.FX.OPERATORYSETUP (OPERATOR = "DERIVATIVE")</i> .....	<b>3-7</b>
EnableAutoScale <i>Bool</i> .....	3-7
FindScale <i>Action</i> .....	3-7
VerOffset <i>Double</i> .....	3-7
VerScale <i>DoubleLockstep</i> .....	3-7
<b>DESKEW</b> <i>APP.MATH.FX.OPERATORYSETUP (OPERATOR = "DESKEW")</i> .....	<b>3-8</b>
WaveDeskew <i>Double</i> .....	3-8
<b>ENVELOPE</b> <i>APP.MATH.FX.OPERATORYSETUP (OPERATOR = "ENVELOPE")</i> .....	<b>3-9</b>
ClearSweeps <i>Action</i> .....	3-9
Sweeps <i>Integer</i> .....	3-9
<b>ERES</b> <i>APP.MATH.FX.OPERATORYSETUP (OPERATOR = "ENHANCEDRESOLUTION")</i> .....	<b>3-10</b>
Bits <i>Enum</i> .....	3-10
<b>EXCELMATH</b> <i>APP.MATH.FX.OPERATORYSETUP (OPERATOR = "EXCELMATH")</i> .....	<b>3-11</b>
AddChart <i>Action</i> .....	3-11
AddLabels <i>Action</i> .....	3-11
Advanced <i>Bool</i> .....	3-12
ClearSheet <i>Action</i> .....	3-12
CreateDemoSheet <i>Action</i> .....	3-12
FindScale <i>Action</i> .....	3-12
NewSheet <i>Bool</i> .....	3-13
OutputCell <i>String</i> .....	3-13
OutputEnable <i>Bool</i> .....	3-13
OutputHeaderCell <i>String</i> .....	3-13
Scaling <i>Enum</i> .....	3-14
Source1Cell <i>String</i> .....	3-14
Source1Enable <i>Bool</i> .....	3-14

## AUTOMATION MANUAL

---

Source1HeaderCell <i>String</i> .....	3-14
Source2Cell <i>String</i> .....	3-14
Source2Enable <i>Bool</i> .....	3-14
Source2HeaderCell <i>String</i> .....	3-15
SpreadsheetFilename <i>FileName</i> .....	3-15
Status <i>String</i> .....	3-15
WithHeader <i>Bool</i> .....	3-16
<b>FFT APP.MATH.FX.OPERATORYSETUP (OPERATOR = "FFT").....</b>	<b>3-17</b>
Algorithm <i>Enum</i> .....	3-17
FillType <i>Enum</i> .....	3-17
SuppressDC <i>Bool</i> .....	3-18
Type <i>Enum</i> .....	3-18
Window <i>Enum</i> .....	3-18
<b>FILTER APP.MATH.FX.OPERATORYSETUP (OPERATOR = "FILTER").....</b>	<b>3-19</b>
AutoLength <i>Bool</i> .....	3-19
CosineBeta <i>Double</i> .....	3-19
CustomFilename <i>FileName</i> .....	3-20
FilterKind <i>Enum</i> .....	3-20
FilterType <i>Enum</i> .....	3-20
FirOrlir <i>Enum</i> .....	3-20
GaussianBT <i>Double</i> .....	3-21
HighFreqPass <i>Double</i> .....	3-21
LowFreqPass <i>Double</i> .....	3-21
NumberOfTaps <i>Integer</i> .....	3-21
PassBandAttenuation <i>Double</i> .....	3-21
PassBandRipple <i>Double</i> .....	3-21
Rolloff <i>Double</i> .....	3-22
StopBandAttenuation <i>Double</i> .....	3-22
TransitionWidth <i>Double</i> .....	3-22
Window <i>Enum</i> .....	3-22
<b>FLOOR APP.MATH.FX.OPERATORYSETUP (OPERATOR = "FLOOR").....</b>	<b>3-23</b>
ClearSweeps <i>Action</i> .....	3-23
Sweeps <i>Integer</i> .....	3-23
<b>HISTOGRAM APP.MATH.FX.OPERATORYSETUP (OPERATOR =</b>	
<b>"HISTOGRAM").....</b>	<b>3-24</b>
AutoFindScale <i>Bool</i> .....	3-24
Bins <i>DoubleLockstep</i> .....	3-24
Center <i>Double</i> .....	3-25
ClearSweeps <i>Action</i> .....	3-25
FindScale <i>Action</i> .....	3-25
HorScale <i>DoubleLockstep</i> .....	3-25
Values <i>Integer</i> .....	3-25
VerScaleType <i>Enum</i> .....	3-25
<b>INTEGRAL APP.MATH.FX.OPERATORYSETUP (OPERATOR =</b>	
<b>"INTEGRAL").....</b>	<b>3-26</b>
Adder <i>Double</i> .....	3-26
FindScale <i>Action</i> .....	3-26
Multiplier <i>Double</i> .....	3-26
VerOffset <i>Double</i> .....	3-27
VerScale <i>DoubleLockstep</i> .....	3-27

## TABLE OF CONTENTS

<b>INTERPOLATE APP.MATH.FX.OPERATORYSETUP (OPERATOR = "INTERPOLATE")</b> .....	<b>3-28</b>
Expand <i>DoubleLockstep</i> .....	3-28
InterpolateType <i>Enum</i> .....	3-28
<b>MATH SCRIPT APP.MATH.FX.OPERATORYSETUP (OPERATOR = "WAVESCRIPT")</b> .....	<b>3-29</b>
Code <i>String</i> .....	3-29
Language <i>Enum</i> .....	3-30
Status <i>String</i> .....	3-30
<b>MATHCADMATH APP.MATH.FX.OPERATORYSETUP (OPERATOR = "MATHCADMATH")</b> .....	<b>3-31</b>
Advanced <i>Bool</i> .....	3-31
FindScale <i>Action</i> .....	3-31
NewSheet <i>Bool</i> .....	3-32
OutputEnable <i>Bool</i> .....	3-32
OutputHeaderVar <i>String</i> .....	3-32
OutputVar <i>String</i> .....	3-33
Reload <i>Action</i> .....	3-33
Scaling <i>Enum</i> .....	3-33
Source1Enable <i>Bool</i> .....	3-33
Source1HeaderVar <i>String</i> .....	3-34
Source1Var <i>String</i> .....	3-34
Source2Enable <i>Bool</i> .....	3-34
Source2HeaderVar <i>String</i> .....	3-34
Source2Var <i>String</i> .....	3-34
Status <i>String</i> .....	3-34
WithHeader <i>Bool</i> .....	3-34
WorksheetFilename <i>FileName</i> .....	3-34
<b>MATLAB MATH APP.MATH.FX.OPERATORYSETUP (OPERATOR = "MATLABWAVEFORM")</b> .....	<b>3-35</b>
MATLABCode <i>String</i> .....	3-35
MATLABPlot <i>Bool</i> .....	3-35
MATLABScalePerDiv <i>Double</i> .....	3-35
MATLABZeroOffset <i>Double</i> .....	3-36
<b>PHISTOGRAM APP.MATH.FX.OPERATORYSETUP (OPERATOR = "PERSISTENCEHISTOGRAM")</b> .....	<b>3-37</b>
CenterCursor <i>Action</i> .....	3-37
ClearSweeps <i>Action</i> .....	3-37
CutDirection <i>Enum</i> .....	3-37
HorCutCenter <i>Double</i> .....	3-38
HorCutWidth <i>Double</i> .....	3-38
PctCutWidth <i>Double</i> .....	3-38
VerCutCenter <i>Double</i> .....	3-38
VerCutWidth <i>Double</i> .....	3-38
<b>PTRACE MEAN APP.MATH.FX.OPERATORYSETUP (OPERATOR = "PERSISTENCETRACEMEAN")</b> .....	<b>3-39</b>
ClearSweeps <i>Action</i> .....	3-39
<b>PTRACE RANGE APP.MATH.FX.OPERATORYSETUP (OPERATOR = "PERSISTENCETRACERANGE")</b> .....	<b>3-40</b>
ClearSweeps <i>Action</i> .....	3-40

**AUTOMATION MANUAL**

PctPopulation <i>Double</i> .....	3-40
<b>PTRACE SIGMA APP.MATH.FX.OPERATORYSETUP (OPERATOR = "PERSISTENCETRACESIGMA")</b> .....	<b>3-41</b>
ClearSweeps <i>Action</i> .....	3-41
Sigma <i>Double</i> .....	3-41
<b>RESCALE APP.MATH.FX.OPERATORYSETUP (OPERATOR = "RESCALE")</b> .....	<b>3-42</b>
Adder <i>Double</i> .....	3-42
CustomUnit <i>Bool</i> .....	3-42
Multiplier <i>Double</i> .....	3-42
Unit <i>String</i> .....	3-43
<b>ROOF APP.MATH.FX.OPERATORYSETUP (OPERATOR = "ROOF")</b> .....	<b>3-44</b>
ClearSweeps <i>Action</i> .....	3-44
Sweeps <i>Integer</i> .....	3-44
<b>SEGMENT APP.MATH.FX.OPERATORYSETUP (OPERATOR = "SEGMENTSELECT")</b> .....	<b>3-45</b>
SelectedSegment <i>Integer</i> .....	3-45
<b>SPARSE APP.MATH.FX.OPERATORYSETUP (OPERATOR = "SPARSE")</b> .....	<b>3-46</b>
SparsingFactor <i>Integer</i> .....	3-46
SparsingPhase <i>Integer</i> .....	3-46
<b>TRACK APP.MATH.FX.OPERATORYSETUP (OPERATOR = "TRACK")</b> .....	<b>3-47</b>
AutoFindScale <i>Bool</i> .....	3-47
Center <i>Double</i> .....	3-47
FindScale <i>Action</i> .....	3-47
VerScale <i>DoubleLockstep</i> .....	3-47
<b>TREND APP.MATH.FX.OPERATORYSETUP (OPERATOR = "TREND")</b> .....	<b>3-48</b>
AutoFindScale <i>Bool</i> .....	3-48
Center <i>Double</i> .....	3-48
ClearSweeps <i>Action</i> .....	3-48
FindScale <i>Action</i> .....	3-48
Values <i>Integer</i> .....	3-48
VerScale <i>DoubleLockstep</i> .....	3-48
<b>MATHCADPARAMARITH APP.MEASURE.PX.OPERATOR (ARITHENGINE = "MATHCADPARAMARITH")</b> .....	<b>3-49</b>
Advanced <i>Bool</i> .....	3-49
NewSheet <i>Bool</i> .....	3-49
OutputEnable <i>Bool</i> .....	3-50
OutputHeaderVar <i>String</i> .....	3-50
OutputVar <i>String</i> .....	3-50
Reload <i>Action</i> .....	3-51
Source1Enable <i>Bool</i> .....	3-51
Source1HeaderVar <i>String</i> .....	3-51
Source1Var <i>String</i> .....	3-52
Source2Enable <i>Bool</i> .....	3-52
Source2HeaderVar <i>String</i> .....	3-52
Source2Var <i>String</i> .....	3-53
Status <i>String</i> .....	3-53
WithHeader <i>Bool</i> .....	3-53
WorksheetFilename <i>FileName</i> .....	3-54

## TABLE OF CONTENTS

<b>P INVERT APP.MEASURE.PX.OPERATOR (ARITHENGINE = "PARAMINVERT")</b> .....	<b>3-55</b>
CycleForTimeUnits <i>Bool</i> .....	3-55
<b>P SCRIPT APP.MEASURE.PX.OPERATOR (ARITHENGINE = "PARAMSCRIPT")</b> .....	<b>3-56</b>
Code <i>String</i> .....	3-56
Language <i>Enum</i> .....	3-56
Status <i>String</i> .....	3-57
Timeout <i>Double</i> .....	3-57
<b>Q APP.MEASURE.PX.OPERATOR (PARAMENGINE = "EYEQ")</b> .....	<b>3-58</b>
PctCutWidth <i>Double</i> .....	3-58
<b>AREA APP.MEASURE.PX.OPERATOR (PARAMENGINE = "AREA")</b> .....	<b>359</b>
Cyclic <i>Bool</i> .....	3-59
<b>AVG POWER APP.MEASURE.PX.OPERATOR (PARAMENGINE = "EYEAVGPOWER")</b> .....	<b>3-60</b>
PctCutWidth <i>Double</i> .....	3-60
<b>DPERIOD@LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "DELTAPERIODATLEVEL")</b> .....	<b>3-61</b>
AbsLevel <i>Double</i> .....	3-61
BaseFrequency <i>Double</i> .....	3-62
FindBaseFrequency <i>Action</i> .....	3-62
FindLevel <i>Action</i> .....	3-62
GroupSize <i>Integer</i> .....	3-63
Hysteresis <i>Double</i> .....	3-63
LevelType <i>Enum</i> .....	3-63
PercentLevel <i>Double</i> .....	3-64
SignalType <i>Enum</i> .....	3-64
Slope <i>Enum</i> .....	3-65
StartCycle <i>Integer</i> .....	3-66
StdBaseFrequency <i>Enum</i> .....	3-66
UseBaseFrequency <i>Enum</i> .....	3-67
<b>DTIME@LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "DELTATIMEATLEVEL")</b> .....	<b>3-68</b>
AbsLevel1 <i>Double</i> .....	3-68
AbsLevel2 <i>Double</i> .....	3-69
FindLevel1 <i>Action</i> .....	3-69
FindLevel2 <i>Action</i> .....	3-69
Hysteresis1 <i>Double</i> .....	3-69
Hysteresis2 <i>Double</i> .....	3-69
LevelType1 <i>Enum</i> .....	3-70
LevelType2 <i>Enum</i> .....	3-70
PercentLevel1 <i>Double</i> .....	3-70
PercentLevel2 <i>Double</i> .....	3-71
Slope1 <i>Enum</i> .....	3-71
Slope2 <i>Enum</i> .....	3-72
<b>DUTY@LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "DUTYATLEVEL")</b> .....	<b>3-73</b>
AbsLevel <i>Double</i> .....	3-73
FindLevel <i>Action</i> .....	3-73
Hysteresis <i>Double</i> .....	3-73

## AUTOMATION MANUAL

---

HysteresisPct <i>Double</i> .....	3-74
HysteresisType <i>Enum</i> .....	3-74
LevelType <i>Enum</i> .....	3-74
PercentLevel <i>Double</i> .....	3-75
Slope <i>Enum</i> .....	3-75
<b>DWIDTH@LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "DELTAWIDTHATLEVEL")</b> .....	<b>3-76</b>
AbsLevel <i>Double</i> .....	3-76
FindLevel <i>Action</i> .....	3-76
Hysteresis <i>Double</i> .....	3-76
LevelType <i>Enum</i> .....	3-76
PercentLevel <i>Double</i> .....	3-76
Slope <i>Enum</i> .....	3-76
<b>EDGE@LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "EDGEATLEVEL")</b> .....	<b>3-78</b>
AbsLevel <i>Double</i> .....	3-78
FindLevel <i>Action</i> .....	3-78
Hysteresis <i>Double</i> .....	3-78
LevelType <i>Enum</i> .....	3-78
PercentLevel <i>Double</i> .....	3-78
Slope <i>Enum</i> .....	3-79
<b>EXCELPARAM APP.MEASURE.PX.OPERATOR (PARAMENGINE = "EXCELPARAM")</b> .....	<b>3-80</b>
AddChart <i>Action</i> .....	3-80
AddLabels <i>Action</i> .....	3-80
Advanced <i>Bool</i> .....	3-80
ClearSheet <i>Action</i> .....	3-80
CreateDemoSheet <i>Action</i> .....	3-80
NewSheet <i>Bool</i> .....	3-81
OutputCell <i>String</i> .....	3-81
OutputEnable <i>Bool</i> .....	3-81
OutputHeaderCell <i>String</i> .....	3-81
Source1Cell <i>String</i> .....	3-81
Source1Enable <i>Bool</i> .....	3-81
Source1HeaderCell <i>String</i> .....	3-81
Source2Cell <i>String</i> .....	3-81
Source2Enable <i>Bool</i> .....	3-81
Source2HeaderCell <i>String</i> .....	3-82
SpreadsheetFilename <i>FileName</i> .....	3-82
Status <i>String</i> .....	3-82
WithHeader <i>Bool</i> .....	3-82
<b>EXT. RATIO APP.MEASURE.PX.OPERATOR (PARAMENGINE = "EXTINCTIONRATIO")</b> .....	<b>3-83</b>
CalcType <i>Enum</i> .....	3-83
PctCutWidth <i>Double</i> .....	3-83
<b>EYE AMPLITUDE APP.MEASURE.PX.OPERATOR (PARAMENGINE = "EYEAMPLITUDE")</b> .....	<b>3-84</b>
PctCutWidth <i>Double</i> .....	3-84
<b>EYE BER APP.MEASURE.PX.OPERATOR (PARAMENGINE = "EYEBER")</b> .....	<b>3-85</b>
PctCutWidth <i>Double</i> .....	3-85

## TABLE OF CONTENTS

<b>EYE HEIGHT APP.MEASURE.PX.OPERATOR (PARAMENGINE = "EYEHEIGHT")</b> .....	<b>3-86</b>
CalcUnits Enum.....	3-86
PctCutWidth Double.....	3-86
<b>FALL@LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "FALLATLEVEL")</b> .....	<b>3-87</b>
HighAbs Double.....	3-87
HighPct Double.....	3-87
LevelsAre Enum.....	3-88
LowAbs Double.....	3-88
LowPct Double.....	3-88
<b>FREQ@LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "FREQUENCYATLEVEL")</b> .....	<b>3-89</b>
AbsLevel Double.....	3-89
BaseFrequency Double.....	3-89
FindBaseFrequency Action.....	3-90
FindLevel Action.....	3-90
Hysteresis Double.....	3-90
LevelType Enum.....	3-90
PercentLevel Double.....	3-90
SignalType Enum.....	3-91
Slope Enum.....	3-91
StdBaseFrequency Enum.....	3-91
UseBaseFrequency Enum.....	3-92
<b>FWXX APP.MEASURE.PX.OPERATOR (PARAMENGINE = "FULLWIDTHATXX")</b> .....	<b>3-93</b>
HFractionHt Double.....	3-93
<b>HALF PERIOD APP.MEASURE.PX.OPERATOR (PARAMENGINE = "HALFPERIOD")</b> .....	<b>3-94</b>
AbsLevel Double.....	3-94
FindLevel Action.....	3-94
Hysteresis Double.....	3-94
LevelType Enum.....	3-94
PercentLevel Double.....	3-94
Slope Enum.....	3-94
<b>HOLD TIME APP.MEASURE.PX.OPERATOR (PARAMENGINE = "HOLDTIME")</b> .....	<b>3-96</b>
ClockAbsLevel Double.....	3-96
ClockFindLevel Action.....	3-96
ClockHysteresis Double.....	3-97
ClockLevells Enum.....	3-97
ClockPctLevel Double.....	3-98
ClockSlope Enum.....	3-98
CursorDisplay Enum.....	3-98
DataAbsLevel Double.....	3-99
DataFindLevel Action.....	3-99
DataHysteresis Double.....	3-100
DataLevells Enum.....	3-100
DataPctLevel Double.....	3-100
DataSlope Enum.....	3-101

## AUTOMATION MANUAL

<b>LEVEL@X APP.MEASURE.PX.OPERATOR (PARAMENGINE = "LEVELATX")</b> .....	<b>3-102</b>
CursorShape Enum .....	3-102
HorValue Double .....	3-102
LevelCursor Bool .....	3-103
Marker Bool .....	3-103
PinToData Bool .....	3-103
TimeCursor Bool .....	3-104
<b>MATHCADPARAM APP.MEASURE.PX.OPERATOR (PARAMENGINE = "MATHCADPARAM")</b> .....	<b>3-105</b>
Advanced Bool .....	3-105
NewSheet Bool .....	3-105
OutputEnable Bool .....	3-105
OutputHeaderVar String .....	3-105
OutputVar String .....	3-105
Reload Action .....	3-106
Source1Enable Bool .....	3-106
Source1HeaderVar String .....	3-106
Source1Var String .....	3-106
Source2Enable Bool .....	3-106
Source2HeaderVar String .....	3-106
Source2Var String .....	3-106
Status String .....	3-107
WithHeader Bool .....	3-107
WorksheetFilename FileName .....	3-107
<b>MATLAB PARAM APP.MEASURE.PX.OPERATOR (PARAMENGINE = "MATLABPARAMETER")</b> .....	<b>3-108</b>
MATLABCode String .....	3-108
MATLABPlot Bool .....	3-108
MATLABScalePerDiv Double .....	3-108
MATLABZeroOffset Double .....	3-108
<b>MAXIMUM APP.MEASURE.PX.OPERATOR (PARAMENGINE = "MAXIMUM")</b> .....	<b>3-109</b>
<b>MEAN APP.MEASURE.PX.OPERATOR (PARAMENGINE = "MEAN")</b> .....	<b>3-110</b>
Cyclic Bool .....	3-110
<b>MEDIAN APP.MEASURE.PX.OPERATOR (PARAMENGINE = "MEDIAN")</b> .....	<b>3-111</b>
Cyclic Bool .....	3-111
<b>MINIMUM APP.MEASURE.PX.OPERATOR (PARAMENGINE = "MINIMUM")</b> .....	<b>3-112</b>
<b>NB PHASE APP.MEASURE.PX.OPERATOR (PARAMENGINE = "NARROWBANDPHASE")</b> .....	<b>3-113</b>
Frequency Double .....	3-113
<b>NB POWER APP.MEASURE.PX.OPERATOR (PARAMENGINE = "NARROWBANDPOWER")</b> .....	<b>3-114</b>
Frequency Double .....	3-114
<b>NPTS APP.MEASURE.PX.OPERATOR (PARAMENGINE = "NPOINTS")</b> .....	<b>3-115</b>
UsePointsInFrame Bool .....	3-115

## TABLE OF CONTENTS

<b>ONE LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "EYONELEVEL")</b> .....	<b>3-116</b>
PctCutWidth <i>Double</i> .....	3-116
<b>PARAM SCRIPT APP.MEASURE.PX.OPERATOR (PARAMENGINE = "PARAMSCRIPT")</b> .....	<b>3-117</b>
Code <i>String</i> .....	3-117
Language <i>Enum</i> .....	3-118
Status <i>String</i> .....	3-118
Timeout <i>Double</i> .....	3-118
<b>PERCENTILE APP.MEASURE.PX.OPERATOR (PARAMENGINE = "PERCENTILE")</b> .....	<b>3-119</b>
HPctPop <i>Double</i> .....	3-119
<b>PERIOD@LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "PERIODATLEVEL")</b> .....	<b>3-120</b>
AbsLevel <i>Double</i> .....	3-120
BaseFrequency <i>Double</i> .....	3-120
FindBaseFrequency <i>Action</i> .....	3-120
FindLevel <i>Action</i> .....	3-121
Hysteresis <i>Double</i> .....	3-121
LevelType <i>Enum</i> .....	3-121
PercentLevel <i>Double</i> .....	3-121
SignalType <i>Enum</i> .....	3-121
Slope <i>Enum</i> .....	3-122
StdBaseFrequency <i>Enum</i> .....	3-122
UseBaseFrequency <i>Enum</i> .....	3-122
<b>PHASE APP.MEASURE.PX.OPERATOR (PARAMENGINE = "PHASE")</b> .....	<b>3-123</b>
OutputType <i>Enum</i> .....	3-123
RefAbsLevel <i>Double</i> .....	3-123
RefFindLevel <i>Action</i> .....	3-124
RefHysteresis <i>Double</i> .....	3-124
RefLevelType <i>Enum</i> .....	3-124
RefPercentLevel <i>Double</i> .....	3-125
RefSlope <i>Enum</i> .....	3-125
SigAbsLevel <i>Double</i> .....	3-125
SigFindLevel <i>Action</i> .....	3-125
SigHysteresis <i>Double</i> .....	3-126
SigLevelType <i>Enum</i> .....	3-126
SigPercentLevel <i>Double</i> .....	3-126
SigSlope <i>Enum</i> .....	3-126
<b>RISE@LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "RISEATLEVEL")</b> .....	<b>3-127</b>
HighAbs <i>Double</i> .....	3-127
HighPct <i>Double</i> .....	3-127
LevelsAre <i>Enum</i> .....	3-127
LowAbs <i>Double</i> .....	3-127
LowPct <i>Double</i> .....	3-128
<b>RMS APP.MEASURE.PX.OPERATOR (PARAMENGINE = "ROOTMEANSQUARE")</b> .....	<b>3-129</b>
Cyclic <i>Bool</i> .....	3-129

**AUTOMATION MANUAL**

<b>SETUP</b>	<b>APP.MEASURE.PX.OPERATOR (PARAMENGINE = "SETUP").....</b>	<b>3-130</b>
	ClockAbsLevel <i>Double</i> .....	3-130
	ClockFindLevel <i>Action</i> .....	3-130
	ClockHysteresis <i>Double</i> .....	3-130
	ClockLevells <i>Enum</i> .....	3-131
	ClockPctLevel <i>Double</i> .....	3-131
	ClockSlope <i>Enum</i> .....	3-131
	CursorDisplay <i>Enum</i> .....	3-131
	DataAbsLevel <i>Double</i> .....	3-131
	DataFindLevel <i>Action</i> .....	3-132
	DataHysteresis <i>Double</i> .....	3-132
	DataLevells <i>Enum</i> .....	3-132
	DataPctLevel <i>Double</i> .....	3-132
	DataSlope <i>Enum</i> .....	3-132
	Summary <i>String</i> .....	3-132
<b>SKEW</b>	<b>APP.MEASURE.PX.OPERATOR (PARAMENGINE = "SKEW").....</b>	<b>3-133</b>
	Clock1AbsLevel <i>Double</i> .....	3-133
	Clock1FindLevel <i>Action</i> .....	3-133
	Clock1Hysteresis <i>Double</i> .....	3-133
	Clock1Levells <i>Enum</i> .....	3-133
	Clock1PctLevel <i>Double</i> .....	3-134
	Clock1Slope <i>Enum</i> .....	3-134
	Clock2AbsLevel <i>Double</i> .....	3-134
	Clock2FindLevel <i>Action</i> .....	3-134
	Clock2Hysteresis <i>Double</i> .....	3-134
	Clock2Levells <i>Enum</i> .....	3-134
	Clock2PctLevel <i>Double</i> .....	3-134
	Clock2Slope <i>Enum</i> .....	3-135
<b>STD DEV</b>	<b>APP.MEASURE.PX.OPERATOR (PARAMENGINE =</b>	
	<b>"STANDARDDEVIATION").....</b>	<b>3-136</b>
	Cyclic <i>Bool</i> .....	3-136
<b>TIE@LEVEL</b>	<b>APP.MEASURE.PX.OPERATOR (PARAMENGINE = "TIE").....</b>	<b>3-137</b>
	AbsLevel <i>Double</i> .....	3-137
	BaseFrequency <i>Double</i> .....	3-137
	CutOffDivisor <i>Double</i> .....	3-137
	DataIsNRZ <i>Bool</i> .....	3-138
	FindBaseFrequency <i>Action</i> .....	3-138
	FindLevel <i>Action</i> .....	3-138
	Hysteresis <i>Double</i> .....	3-138
	IncludeVirtualEdges <i>Bool</i> .....	3-138
	LevelType <i>Enum</i> .....	3-138
	PercentLevel <i>Double</i> .....	3-138
	ResultScaling <i>Enum</i> .....	3-139
	SignalType <i>Enum</i> .....	3-139
	Slope <i>Enum</i> .....	3-139
	StdBaseFrequency <i>Enum</i> .....	3-139
	UseBaseFrequency <i>Enum</i> .....	3-140
	UseGoldenPLL <i>Bool</i> .....	3-140
<b>TIME@LEVEL</b>	<b>APP.MEASURE.PX.OPERATOR (PARAMENGINE =</b>	
	<b>"TIMEATLEVEL").....</b>	<b>3-141</b>
	AbsLevel <i>Double</i> .....	3-141
	FindLevel <i>Action</i> .....	3-141
	Hysteresis <i>Double</i> .....	3-141

---

**T A B L E O F C O N T E N T S**


---

HysteresisPct <i>Double</i> .....	3-141
HysteresisType <i>Enum</i> .....	3-141
LevelType <i>Enum</i> .....	3-142
PercentLevel <i>Double</i> .....	3-142
Slope <i>Enum</i> .....	3-142
<b>WIDTH@LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "WIDTHATLEVEL")</b> .....	<b>3-143</b>
AbsLevel <i>Double</i> .....	3-143
FindLevel <i>Action</i> .....	3-143
FirstWhenBoth <i>Enum</i> .....	3-143
Hysteresis <i>Double</i> .....	3-144
HysteresisPct <i>Double</i> .....	3-144
HysteresisType <i>Enum</i> .....	3-144
LevelType <i>Enum</i> .....	3-144
PercentLevel <i>Double</i> .....	3-144
Slope <i>Enum</i> .....	3-144
<b>X AT PEAK APP.MEASURE.PX.OPERATOR (PARAMENGINE = "XATPEAK")</b> .....	<b>3-145</b>
PeakNumber <i>Integer</i> .....	3-145
<b>ZERO LEVEL APP.MEASURE.PX.OPERATOR (PARAMENGINE = "EYEZEROLEVEL")</b> .....	<b>3-146</b>
PctCutWidth <i>Double</i> .....	3-146

BLANK PAGE

## I N T R O   D U C T I O N

## About this Manual

This manual is a reference guide to the “Automation” capabilities of LeCroy’s X-Stream™ DSOs.

The manual includes a complete list of all instrument controls that are available to a controlling application. In contrast to previously available remote control possibilities for LeCroy instruments, Automation enables the controlling application to run on the instrument itself.

Part One, “About Automation,” covers the principles of control via Automation and offers practical examples. It also presents the overall hierarchy of controls accessible via Automation.

Part Two, “Control Reference,” presents each of the controls in detail, including examples of use in many cases.

Part Three, “Math/Measure Control Reference,” presents controls available in each of the multitude of the Math/Measure ‘processors’ available in X-Stream DSOs.

The logo for X-STREAM, featuring the word "X-STREAM" in a bold, italicized, sans-serif font. The "X" is red, and the rest of the letters are black.

BLANK PAGE

# ABOUT AUTOMATION

This section presents an overview of the Automation capabilities of X-Stream DSOs along with an overview of the technology itself.

## PART ONE: ABOUT AUTOMATION

---

### CHAPTER ONE: *Overview*

In this chapter learn about

- *Microsoft's COM-based "OLE Automation"*
- *How to create simple applications in Visual Basic to control the instrument*
- *How to use the X-Stream Browser to interactively control the instrument*

---

## OVERVIEW OF AUTOMATION

In addition to supporting the familiar ASCII-based remote commands that have been used to control all LeCroy DSOs for many years, all of the Windows-based “X-Stream” instruments fully support control by Automation interfaces based on Microsoft’s Component Object Model (COM). Using COM, the controlling application can run directly on the instrument without requiring an external controller; or, alternatively, it can run using Microsoft’s distributed COM standard (DCOM) on a networked computer.

### Standards

Automation is a Microsoft technology, formerly referred to as “OLE Automation,” that has primarily been used to enable cross-application macro programming. It is based upon the Component Object Model (COM), which is similar in nature to CORBA, more commonly found in the UNIX world.

An application that “exposes Automation Objects” is referred to as an “Automation Server.” Automation Objects expose “Automation Interfaces” to the controlling “Automation Client.” This manual describes these Automation objects and interfaces in detail.

It is important to note that Automation itself is not language dependent; it can be used from any programming language that supports COM. This manual, however, concentrates mainly on the use of Automation from the Visual Basic Script (VBScript) language for several reasons, including the fact that it is one of the easiest to use. Also, it is the language that X-Stream instruments use for setup files (more on this later). In addition, the VBScript interpreter is installed by default on all X-Stream instruments and, therefore, is available without your having to purchase any additional software.

### Compatibility with Other LeCroy Scopes

Throughout LeCroy’s history, we have striven to maximize compatibility, and this policy remains in force. However, due to the fact that the underlying technologies used by Automation require the 32-bit Windows operating system, and that this system is available only on our X-Stream instruments, Automation is not available on the older LeCroy scope families.

### Automation and IEEE 488.2 Remote Control – How Do They Compare?

Automation does not replace the IEEE 488.2-based remote command set, which is also supported by X-Stream instruments (and will continue to be). Instead, it augments it and allows a new class of application to be created that can run on the DSO itself.

Automation however can be considered as the “Native Language,” or “Mother Tongue” of X-Stream instruments. All of the instrument’s controls and features are available to the Automation Client.

Automation controls generally are more granular than 488.2 remote commands. That is, many 488.2 remote commands set more than one control at the same time; whereas, via Automation, this is not the case.

## PART ONE: ABOUT AUTOMATION

The following table summarizes the differences between the two remote control possibilities:

	IEEE 488.2 Control	Automation Control
<b>Physical Transport</b>	GPIB, TCP/IP over Ethernet	Inter-process using COM, inter-PC using DCOM (TCP/IP)
<b>Textual parsing of instrument responses required</b>	Yes. All instrument responses need 'parsing' to extract useful information.	No. Each element in the Automation hierarchy appears as a "variable" to the Automation client.
<b>Compatibility with previous LeCroy DSOs</b>	Very good. In most cases remote control applications written for older DSOs will work without modification.	None. Automation is a new standard first introduced with LeCroy's X-Stream DSOs.
<b>Ability to run controlling application "in the box"</b>	Yes, by using the TCP/IP (VICP) protocol to talk to the "local host"	Yes, natively
<b>Ease of Use</b>	Not trivial. It's easier using a tool such as ActiveDSO that hides some of the complexities.	Very easy to use from scripting languages and office productivity tools
<b>Format of Waveform results</b>	Binary or ASCII. Both require parsing before use.	Waveforms are presented as arrays of floating point values .
<b>Control from MS Office suite</b>	Possible via ActiveDSO utility	Yes, natively (see examples later in this manual)

### General Characteristics

- When an application is running locally on the instrument and requests a connection to the DSO via Automation (for example, by using **CreateObject("LeCroy.XStreamDSO")** from Visual Basic), one of two things will happen. If the X-Stream DSO application is already running, the object returned will be a "pointer" to the running application. If the DSO application is not running, it will be started. It is not possible to run two simultaneous instances of the X-Stream DSO application.
- More than one simultaneous connection to the instrument via Automation will be accepted, but simultaneous connections are not recommended.
- When the final client has been disconnected from the instrument (server), the X-Stream DSO application will remain running and will accept further client connections.
- Operations that cause Modal Dialogs to appear in the instrument's display will, by default, disrupt access from Automation. This behavior can be changed using the **app.SystemControl.ModalDialogTimeout** and **app.SystemControl.EnableMessageBox** controls. Refer to the description of each of these controls in the reference section for more information.
- The instrument application can be minimized in order to allow the controlling application to take over the display and touch panel by means of the **app.Minimize** control. It can also be resized and repositioned on the display by means of the **app.Top**, **app.Left**, **app.Bottom**, **app.Right** controls.

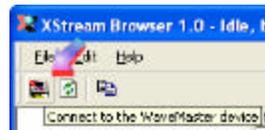
## INTRODUCTION TO THE X-STREAM BROWSER

The easiest way to get up and running with Automation, and also to visualize the “X-Stream Object Model” is to use the X-Stream browser tool, which is pre-installed on all instruments.

To launch the tool, first minimize the instrument application (**File->Minimize**), then double-click on the X-Stream Browser icon on the desktop:



To connect to the running DSO application, click the **Connect** toolbar icon:



Upon connection, the root of the object hierarchy is shown in a layout similar to that presented when a file system is browsed using Windows Explorer.

As a quick demonstration of how the X-Stream Browser can be used, open the **Acquisition** folder, then click on the **C1** folder. Find the **VerScale** (Volts/Div) control in the right-hand window and right-click on it, then select the **Set Value** menu option.

Name	Value	Type
AverageSweeps	1	Integer
AxisRotation	45	Integer
AxisYRotation	20	Integer
BandwidthLimit	Full	Enum
ClearSweeps		Action
Coupling	DC50	Enum
Deskew	0	Double
InterpolateType	Linear	Enum
Invert	false	Bool
LabelsPosition		String
LabelsText		String
NumSegmentsDisplayed	1	Integer
Persist3DQuality	Solid	Enum
Persisted	false	Bool
Persistence3d	false	Bool
PersistenceMonoChrome	true	Bool
PersistenceSaturation	50	Integer
PersistenceTime	Infinite	Enum
ProbeAttenuation	1	Double
SegmentMode	Adjacent	Enum
ShowLastTrace	false	Bool
StartSegment	1	Integer
UseDotJoin	true	Bool
UseGrid	YT1	String
VerOffset	0	Double
<b>VerScale</b>	<b>0.1</b>	<b>DoubleLockstep</b>
VerScaleVariable	false	Bool
View	true	Bool
ViewLabels	false	Bool

Context Menu Options:

- Copy Path...
- Refresh...
- Set Value...**
- Numeric::Replace the numeric value

Set Control Variable value... Dialog:

Information

Name: VerScale

Type: DoubleLockstep      Flags: FB

Range: From 0.002 to 1 step 0.001, locked to 1 2 5

Value:       Set this value.

Enter a new value for **VerScale** and click the **Set this value** button. Restore the DSO's X-Stream window and note that the V/Div value for C1 should have changed to reflect the entered value.

## PART ONE: ABOUT AUTOMATION

STEP-BY-STEP INTRODUCTION TO AUTOMATION USING  
VBScript

This section of the manual presents a walk-through of how to create a simple remote control application, which will run on the instrument, from scratch. It doesn't rely upon any 3<sup>rd</sup>-party development tools, since it uses Windows' built-in text editor (Notepad) and the Visual Basic Script interpreter (VBScript), which is also installed on all instruments.

1. Use the **File? Windowed** menu option to place the DSO application into windowed mode. This allows the windows start-bar to be accessed.
2. Open Windows **Notepad** via Start->Programs->Accessories ->Notepad
3. Write the following text into the editor:

```
Set app = CreateObject("LeCroy.XStreamDSO")
app.AutoSetup
app.Display.GridMode = "Quad"
myVerScale = app.Acquisition.C1.VerScale
MsgBox myVerScale
```

4. Save the file to drive **D:\** and name it **Exercise1.vbs**. Leave Notepad open, we'll need it again.
5. Open Windows Explorer, via **Start? Programs? Accessories? Windows Explorer**.
6. Navigate to drive **D:\** and double-click on **Exercise1.vbs**.
7. That's it. If these steps were followed correctly, you should hear relays clicking while the scope performs an auto-setup operation and enters its quad-grid display mode.

So, what did this "program" actually do?

- The **CreateObject** statement.

```
Set app = CreateObject("LeCroy.XStreamDSO")
```

CreateObject is the Visual Basic function that creates an instance of a COM Server (a.k.a. ActiveX Control). The argument "**LeCroy.XStreamDSO**" refers to our DSO application. Once it has instantiated (connected to) our DSO application we need some kind of 'handle' (pointer) to it so that we can use it later to communicate with the instrument. **CreateObject** returns a handle to us, which we store in the **app** variable.

**NOTE: Only a single instance of the X-Stream DSO software can run on a system at one time. If the DSO software is already running when CreateObject is called, a handle to that running instance is returned. If the DSO software is not running, it will be started.**

- The **app.AutoSetup** statement.

```
app.AutoSetup
```

Using the **app** handle, this line of code calls the **AutoSetup** method, which performs the same task as the front-panel Auto-Setup button. Documentation for this method can be found later in the reference section.

- The **app.Display.GridMode = "Quad"** statement.

```
app.Display.GridMode = "Quad"
```

Using the **app** handle, this line of code sets the **GridMode** control of the **Display** system to the value "Quad". It's important to note that the controls are arranged in a hierarchy, with each 'level' of the hierarchy delimited with a decimal point ( . ).

- The **myVerScale = app.Acquisition.C1.VerScale** statement.

```
myVerScale = app.Acquisition.C1.VerScale
```

Instead of setting the value of a control, this line of code retrieves the current value of a control, in this case the Vertical Scale (Volts/Div) of Channel 1. The value returned is stored within the variable **myVerScale**.

**NOTE: In Visual Basic Script it is not necessary to "Dimension" variables before using them (for example, using statements like "Dim myVerScale as Double").**

- The **MsgBox myVerScale** statement

```
MsgBox myVerScale
```

This line of code does not communicate with the scope at all, but calls the standard Visual Basic Script function **MsgBox**. This function displays a dialog containing the value of the variable following "MsgBox". In our case the value of Channel 1's vertical scale, and waits for the **OK** button to be clicked.

Documentation about the MsgBox function can be found in Microsoft's Visual Basic Scripting documentation at [www.microsoft.com/scripting](http://www.microsoft.com/scripting).

Another point that should be mentioned here is something that is used extensively in Setup files created by the instrument: the ability to use "abbreviations" to simplify programs. Following is an example in which a shorthand method is used to replace some rather long-winded code. It is also important to note that this also enhances performance. For example, the "lookup" of the object `app.Acquisition.C1` occurs only once in the modified code, but three times in the original code.

Instead of:

```
app.Acquisition.C1.VerScale = 0.5
app.Acquisition.C1.VerOffset = 0.1
app.Acquisition.C1.Coupling = "DC50"
```

The following may be used:

```
set myChannell = app.Acquisition.C1
myChannell.VerScale = 0.5
myChannell.VerOffset = 0.1
myChannell.Coupling = "DC50"
```

## **PART ONE: ABOUT AUTOMATION**

---

### **WHERE IS AUTOMATION USED?**

Automation is used in several places in the X-Stream based instrument.

- Instrument Setups (Panel Files)
- Custom Math/Measurements
- CustomDSO, User Interface customization
- Control from external applications (COM/DCOM)

Each of these uses is described in more detail in the following sections.

## SETUPS (PANEL FILES) ARE PROGRAMS!

Setup files, used to save and recall the state of the instrument between runs, are traditionally binary files whose internal structure is neither documented nor obvious to the user.

In XStream DSOs, however, this is no longer the case. Setups are ASCII text files that contain a complete Visual Basic Script “program” that, when “executed,” will restore the instrument to a previously recorded state. **In effect, each time a panel is saved, the instrument effectively writes you a program that, when executed, returns the instrument to the saved state.**

Due to the fact that these setups are already programs, they are a great way to get started quickly with Automation. As an example, try saving a setup into a file and examine it using a text-editor, as follows:

1. Touch **File** in the menu bar, then **Save Setup** from the drop-down menu.
2. Touch the **Browse** button and specify drive **D:** as the location to save the .Iss (LeCroy setup script) file.



3. Touch **Save Now!**
4. Minimize the application using the **Minimize** option from the **File** menu
5. Open Microsoft's Notepad application from the **Accessories** program folder (**Start? Programs? Accessories**).
6. Open the file saved above. You will see a Visual Basic Script program that begins like this:

```
' XStreamDSO ConfigurationVBScript ...
' LECROY,WM8300,WM000001,0.0.0
' Thursday, February 20, 2003 11:26:55 AM

On Error Resume Next
set XStreamDSO = CreateObject("LeCroy.XStreamDSO")
XStreamDSO.RecallingSetup = True

' AladdinPersona ...
XStreamDSO.HideClock = False
XStreamDSO.TouchScreenEnable = True
...
```

Since the entire state of the instrument, including all controls for all installed software options, is saved, this panel may look fairly complex. But don't let this fool you; the basic concept is, in fact, fairly simple.

As a quick example of how setups can be used as the starting point for controlling applications, scroll down to the end of the file and add the following code (shown in bold-type) to the file.

```
' Place any custom VBScript code after this point
'
' Perform an Auto Setup
XStreamDSO.AutoSetup
```

Add this  
code

When this setup is recalled, the complete state of the instrument will be restored, followed by an Auto-Setup operation.

Obviously this is a fairly trivial “application,” but it is easy to imagine how automated testing could be performed with the introduction of loops and conditional execution.

## PART ONE: ABOUT AUTOMATION

---

**NOTE:** Setup files stored by the instrument have file extension “.lss” (LeCroy Setup Script). These files are syntactically identical to Microsoft Visual Basic Script (VBScript) files, which have a “.vbs” extension.



**TIP:** A simple alternative to recalling the panel into the instrument is to execute it, either by double-clicking on the .lss file in Windows Explorer, or by executing it from the command line.



---

## CUSTOM MATH AND MEASUREMENTS

Custom Math and Measurements can be coded using VBScript or JavaScript in instruments equipped with the XDEV and/or XMAP software options. Using Automation control of the instrument, decisions can be made during custom processing that reconfigure the DSO.

When you are developing custom processing routines using the reference section of this manual, **app.Acquisition.Cx.Out.Result** may be used as a comprehensive reference to the **Result Object**, which is used to describe waveform data (InResult, InResult1, InResult2, OutResult).

For more detail about this capability, see the “Customization” section of the on-line Help Manual.

### CustomDSO

CustomDSO enables customization of the instrument’s UI in instruments equipped with the XDEV and/or XMAP options. Two modes of operation are supported: Basic mode and Plug-in mode.

In Basic mode a Visual Basic Script (VBScript) program can be assigned to each of 8 buttons that can, optionally, appear at the bottom of the instrument’s display. By means of Automation, each of these may further reconfigure all 8 buttons, which would allow simple menu hierarchies to be generated.

In Plug-in mode an ActiveX control, created in any of a number of programming languages, can be inserted into the instrument’s menu system. Once “embedded,” this Plug-in can take full control of the DSO using Automation.

Full documentation on CustomDSO is available in the CustomDSO section of the on-line Help Manual.

# PART ONE: ABOUT AUTOMATION

## Control from External Applications

Control of an X-Stream based instrument by Automation is possible from most modern programming languages (interpreted and/or compiled), and also from the "macro" capability of office productivity suites such as Microsoft Office.

The image is a collage of software screenshots demonstrating automation. At the top left is the 'About MATLAB' window showing 'Version 6.5.0.180013a Release 13 June 18, 2002'. To its right is the 'New Project' dialog for 'Microsoft Visual Basic'. In the center is a LeCroy oscilloscope measurement window showing a sine wave and a measurement table with columns for P1 through P8. Below this is a screenshot of Microsoft Excel 2000 showing a spreadsheet with numerical data and a histogram. At the bottom left is the 'Mathcad 2001i' logo and a snippet of a Mathcad worksheet with mathematical formulas like  $f(x) = \sin(x)$  and  $\nabla(\alpha\beta) = \alpha\nabla\beta$ . A yellow box labeled 'Memory' with 'M1' and 'M2 and M3 not shown' is overlaid on the MATLAB window.

**From Visual Basic**

From Visual Basic the **CreateObject** method is used to create the connection to an instrument by Automation.

The following code example creates this connection and sets up some of the instrument's controls:

```
` Connect to the X-Stream DSO
Dim app as Object
Set app = CreateObject("LeCroy.XStreamDSO")

` Setup Vertical and Horizontal settings
app.Acquisition.C1.VerScale = 0.5
app.Acquisition.C1.VerOffset = 0.25

app.Acquisition.Horizontal.HorScale = 0.000001

` Disconnect from the DSO
Set app = Nothing
```

**From MATLAB**

MATLAB uses the **actxserver** keyword to connect to the instrument.

The following code example creates this connection, enables variable vertical scale, reads the vertical scale value for C1, and increases it by a factor of 0.75.

```
DSO = actxserver('Lecroyxstreamdso')
set(DSO.Acquisition.C1.VerScaleVariable,'value',-1)
verscale = get(DSO.Acquisition.C1.VerScale,'value')
verscale = 0.75 * verscale
set(DSO.Acquisition.C1.VerScale,'value',verscale)
```

**NOTE:** Don't confuse the control of the instrument from MATLAB (MATLAB "drives") with the use of MATLAB from within a custom processing function (the instrument "drives").

## PART ONE: ABOUT AUTOMATION

---

### From MS Office (Excel)

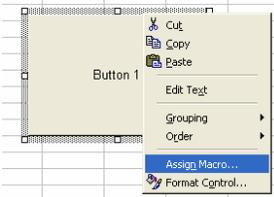
Using Automation, control of the instrument from Microsoft Excel is very similar to control from Visual Basic. This is because the “macro language” used in the office suite is Visual Basic for Applications, a lightweight version of Visual Basic.

The following example shows how to add a button to an Excel spreadsheet that connects to, and controls, the instrument on which Excel is running. Note that this example was generated using Excel 2000, other versions of Excel support similar functionality, but the specific sequence of commands may be slightly different:

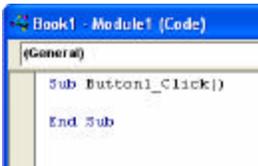
1. Enable the Forms toolbar using the **View? Toolbars? Forms** menu option.



2. Click the button icon  on the toolbar and draw a button on the spreadsheet. The button will be labeled “Button 1” by default.
3. Right-click on the edge of the button and select **Assign Macro** from the drop-down menu.



4. Select the **New** button from the **Assign Macro** dialog; the macro editor will appear:



5. Type the following code into the subroutine:

```
Set app = CreateObject("LeCroy.XStreamDSO")
app.AutoSetup
Set app = Nothing
```

Clicking on the newly created button will now execute this code segment, which connects to the DSO and performs an Auto Setup.

## CONTROL BY DISTRIBUTED COM (DCOM)

Distributed COM (DCOM) can be used to control an X-Stream based instrument remotely over any network supporting TCP/IP (including the Internet).

For obvious security reasons all LeCroy instruments are shipped with this feature disabled, but the fairly simple procedure detailed below can be followed to enable DCOM and define network user accounts that have the access permissions to use it.

There are two basic ways to configure DCOM (User-Level or Share-Level), the choice of which really depends upon whether the client and server (controlling PC and instrument) are both members of the same NT Domain or not.

### Security Settings on the Instrument (Server): User Level

This mode requires that the client and server be logged into the same NT Domain. Use the share-level configuration (below) if this is not the case.

1. Run **DCOMCnfg.exe** from either **Start? Programs? Accessories? Command Prompt** or from **Start? Run...**
2. Select **LeCroyXStreamDSO** from the list of applications.
3. Click the **Properties** button.
4. Click the **General** tab and set the Authentication Level control to **Connect**.
5. Click the **Security** tab. Select **Use custom access permissions** and add the list of users that are allowed to control this scope via DCOM. Select **Everyone** (All Users) if you wish everyone to have access.
6. Click the **Identity** tab and select **The Interactive User**.

### Security Settings on the Instrument (Server): Share Level

1. Run **DCOMCnfg.exe** from either **Start? Programs? Accessories? Command Prompt** or from **Start? Run...**
2. Select **LeCroyXStreamDSO** from the list of applications.
3. Click the **Properties** button.
4. Click the **General** tab and set the Authentication Level control to **(None)**.
5. Click the **Security** tab. Select **Use custom access permissions** and add **Everyone** to the list.
6. Click the **Identity** tab and select **The interactive user**.

### Initialize the Controlling PC (Client)

1. Copy **LeCroyXStreamDSO.exe** from the **C:\Program Files\LeCroy\XStream** directory onto the controlling PC (location not important).
2. Execute **LeCroyXStreamDSO.exe**. Note that it will not run, but it will install enough information into the registry to allow remote control of other instruments.

### Connecting to a Remote Instrument

1. **Visual Basic:** Add a second argument to the **CreateObject** method, which specifies the network location of the remote instrument:

```
Set o = CreateObject("LeCroy.XStreamDSO.1", "wavemaster00121")
```

## PART ONE: ABOUT AUTOMATION

---

o.AutoSetup

2. **MATLAB:** Add a second argument to the actxserver call:

```
h = actxserver (progid [, MachineName])
```

Once connection is made using one of the above techniques, the remainder of the communication with the instrument is the same as it would be in the case where the application runs directly on the instrument.

Remember, however, that because the connection is via a network, the performance of DCOM control of an instrument will not equal that of a direct connection created by running the client application directly on the instrument.

**NOTE: Use version 2.6.0.0 or later for DCOM remote operation.**



## CONTROL VARIABLES EXPLAINED

Traditionally, properties presented to an Automation Client are simple “variables” with types such as Integer (int), String (BSTR), Floating Point (single, double), etc.

Control variables in X-Stream are an extension of the traditional Automation pattern, without affecting how they appear to most Automation clients (see section below on early/late bound clients).

As an example of what this enables, consider the following:

Take a control such as **VerScale** (Volts/Div), this may be set and queried in Visual Basic as follows:

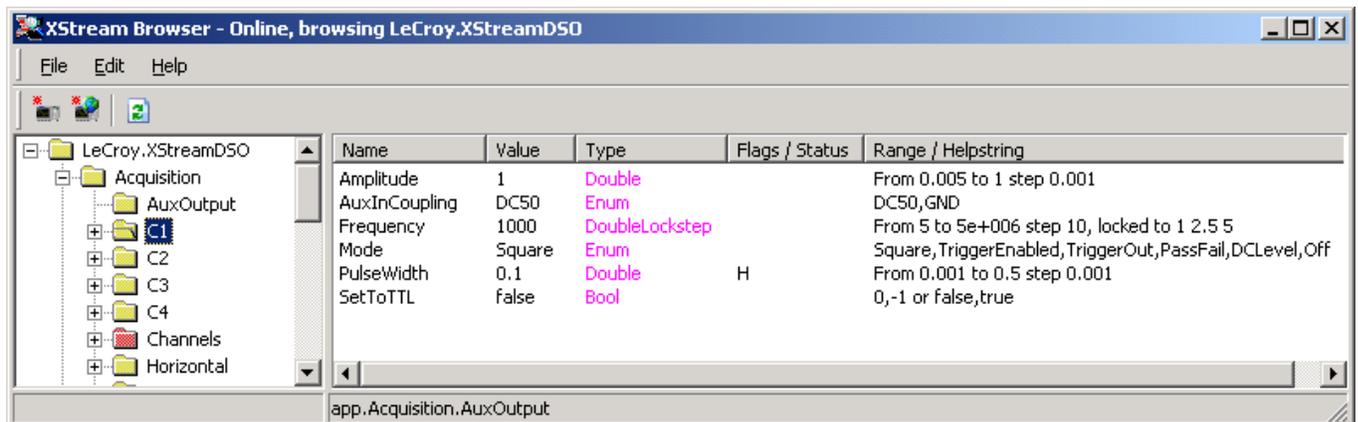
```
app.Acquisition.C1.VerScale = 0.5
CurrentValue = app.Acquisition.C1.VerScale
```

In addition, the following is supported:

```
minValue = app.Acquisition.C1.VerScale.GetMinValue
maxValue = app.Acquisition.C1.VerScale.GetMaxValue
```

This enables an Automation Client to not only set and query the current value of a control, but also to query its limits. This is useful in the generation of instrument-independent applications, or applications that present scope controls in a graphical user interface in which limits are required.

Various types of control variables are supported. The **Type** column in the X-Stream browser shows the control type:



The type designations are also given in the reference section of this manual, and are defined as follows:

<b>Integer</b>	32-bit Integer Value
<b>Double</b>	Double-precision floating point value
<b>DoubleLockstep</b>	Double-precision floating point value locked to a non-linear (e.g., 1, 2, 5) sequence.
<b>Enum</b>	List Value (e.g., “Orange,” “Apple,” “Pear”)
<b>String</b>	String value
<b>Bool</b>	Boolean Value { True, False }, { 0, -1 }
<b>Action</b>	Action (no arguments or value)

## PART ONE: ABOUT AUTOMATION

---

The properties and methods available for each control are type-specific. Listed below are the most commonly used:

TYPE	PROPERTIES
<b>Integer</b>	VARIANT Value Value(VARIANT) int GetAdaptedValue SetRequestedValue(int) int GetRequestedValue int GetDefaultValue int GetGrain int GetMax int GetMin Increment(int)
<b>Double</b>	VARIANT Value Value(VARIANT) double GetAdaptedValue SetRequestedValue(double) double GetRequestedValue double GetDefaultValue double GetGrainValue double GetMaxValue double GetMinValue Increment(int)
<b>DoubleLockstep</b>	See Double
<b>Enum</b>	VARIANT Value Value(VARIANT) int GetAdaptedValue SetRequestedValue(int) int GetRequestedValue int GetDefaultValue int GetMax int GetMin int GetNumberOfValueStates Increment(int) BSTR GetRangeStringScreen BSTR GetRangeStringRemote
<b>String</b>	VARIANT Value Value(VARIANT) int GetMaxLength BSTR GetRequestedValue BSTR GetAdaptedValue SetRequestedValue(BSTR)
<b>Bool</b>	VARIANT Value Value(VARIANT) BOOL GetAdaptedValue BOOL GetRequestedValue BOOL GetDefaultValue Set Clear
<b>Action</b>	ActNow

## ACCESSING WAVEFORM/MEASUREMENT RESULTS

### Waveforms

Waveform data is exposed by a 'Result' object, which appears at various places in the object hierarchy depending upon which waveform is to be accessed. Some examples follow:

```
app.Acquisition.C1.Out.Result
app.Math.F1.Out.Result
app.Memory.M1.Out.Result
```

Waveform data is exposed as a simple array, no deciphering of proprietary binary formats is performed, as was necessary in the past. An example of how it is used follows.

The example is coded as an Excel macro, and should be assigned to a button as described earlier. The macro reads the number of samples in the waveform and places it in cell B1 of the Excel spreadsheet. It then reads all available sample data values and copies them into cells in the first column of the spreadsheet (A1...Axx).

```
Sub Button1_Click()
    ' Connect to the DSO
    Set app = CreateObject("LeCroy.XStreamDSO")

    ' Query the number of samples in C1 and store in cell "B1"
    numSamples = app.Acquisition.C1.Out.Result.Samples
    Cells(1, 2).Value = numSamples

    ' Access the waveform data array, and fill the first column
    ' of the spreadsheet with it
    wave = app.Acquisition.C1.Out.Result.DataArray
    For i = 0 To numSamples - 1
        Cells(i + 1, 1).Value = wave(i)
    Next i
End Sub
```

**NOTE:** Ensure that the record length is < 32kSamples, since Excel has a limit on the number of rows in a spreadsheet. Ideally, you should start experimenting with short (500 point) records.

## PART ONE: ABOUT AUTOMATION

### Measurements

Measurement results are read in the same way as Waveforms. The following example, when copied into an Excel macro, will enable Standard Vertical parameters. It will then transfer the eight parameter values into the spreadsheet (cells C1...C8):

```
Sub Button1_Click()
    ' Connect to the DSO
    Set app = CreateObject("LeCroy.XStreamDSO")

    ' Enable Standard Vertical Parameters
    app.Measure.MeasureMode = "StdVertical"

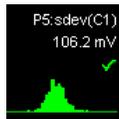
    ' Transfer the 8 parameter values into the spreadsheet
    Cells(1, 3).Value = app.Measure.P1.Out.Result.Value
    Cells(2, 3).Value = app.Measure.P2.Out.Result.Value
    Cells(3, 3).Value = app.Measure.P3.Out.Result.Value
    Cells(4, 3).Value = app.Measure.P4.Out.Result.Value
    Cells(5, 3).Value = app.Measure.P5.Out.Result.Value
    Cells(6, 3).Value = app.Measure.P6.Out.Result.Value
    Cells(7, 3).Value = app.Measure.P7.Out.Result.Value
    Cells(8, 3).Value = app.Measure.P8.Out.Result.Value

    Set app = Nothing
End Sub
```

Statistics are also available for each parameter:

```
app.Measure.P1.Statistics("mean").Result
app.Measure.P1.Statistics("max").Result
app.Measure.P1.Statistics("min").Result
app.Measure.P1.Statistics("num").Result
app.Measure.P1.Statistics("sdev").Result
```

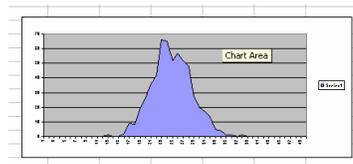
In addition, the data used to display the Histogram is available using the "histo" statistic:



```
Sub Button1_Click()
    Set app = CreateObject("LeCroy.XStreamDSO")

    bins = app.Measure.P5.Statistics("histo").Result.BinPopulations
    numBins = app.Measure.P5.Statistics("histo").Result.bins
    For i = 0 To numBins - 1
        Cells(i + 1, 4).Value = bins(i)
    Next i

    Set App = Nothing
End Sub
```



## Result Status

The waveform result object described above includes a status property (bit-field) that reflects the current status of the trace. This includes both 'warning' and 'error' conditions, as described below.

Description	Value	Bit #
LEC_Valid	0x0	N/A
LEC_Invalid	0x0000000000000001	0
LEC_Overflow	0x0000000000000002	1
LEC_Underflow	0x0000000000000004	2
LEC_ContainsUndefinedValues	0x0000000000000008	3
LEC_LessThan	0x0000000000000010	4
LEC_GreaterThan	0x0000000000000020	5
LEC_NotAPulse	0x0000000000000040	6
LEC_NotCyclic	0x0000000000000080	7
LEC_Averaged	0x0000000000000100	8
LEC_UnlockedPLL	0x0000000000000200	9
LEC_OtherError	0x0000000000000400	10
LEC_OtherWarning	0x0000000000000800	11
LEC_OtherInfo	0x0000000000001000	12
LEC_InputsIncompatible	0x0000100000000000	28
LEC_AlgorithmLimitsReached	0x0000200000000000	29
LEC_BadDefinition	0x0000400000000000	30
LEC_TooFewData	0x0000800000000000	31
LEC_TooManyData	0x0001000000000000	32
LEC_UniformHorizIntervalRequired	0x0002000000000000	33
LEC_BadUnits	0x0004000000000000	34
LEC_DataRangeTooLow	0x0008000000000000	35
LEC_DataUndersampled	0x0010000000000000	36
LEC_PoorStatistics	0x0020000000000000	37
LEC_SlowTransitionTime	0x0040000000000000	38
LEC_DataResampled	0x0080000000000000	39
LEC_DataInterpolated	0x0100000000000000	40
LEC_MeasurementScaleImprecise	0x0200000000000000	41
LEC_NoDataAvailable	0x0400000000000000	42
LEC_SomeCumulatedResultsInvalid	0x0800000000000000	43
LEC_InsufficientMemory	0x1000000000000000	44
LEC_ChannelNotActive	0x2000000000000000	45
LEC_UseStatusDescription	0x4000000000000000	46

## PART ONE: ABOUT AUTOMATION

---

### SYNCHRONIZATION

Synchronization or, more specifically, knowing when to read results, is critical when working with a digital oscilloscope by remote control (it is just as important by IEEE488.2 control as by Automation). This is especially true when working with an oscilloscope that uses a multithreaded architecture.

A classic problem seen in the majority of custom applications that control LeCroy (or other) DSOs is that the scope is left to free-run in Auto-trigger mode while simultaneously (and asynchronously) results are queried.

While working with the instrument via the Automation interface, there are a few techniques that can be used to guarantee the synchronization and consistency of results, whether they be waveform or parameter measurements.

The following example demonstrates a useful technique for ensuring synchronization. This example runs as an Excel macro:

```
Sub Button1_Click()  
    ' Connect to the DSO  
    Set app = CreateObject("LeCroy.XStreamDSO")  
  
    ' Enable Standard Vertical parameters  
    app.Measure.MeasureMode = "StdVertical"  
  
    ' Stop the free-running trigger and take a single acquisition  
    ' Use a 10 second timeout in the case that a trigger is not detected  
    app.Acquisition.TriggerMode = "Stopped"  
    app.Acquisition.Acquire (10, True)  
  
    ' Read the first parameter value and transfer into the spreadsheet  
    Cells(1, 3).Value = app.Measure.Pl.Out.Result.Value  
  
End Sub
```

The **Acquire** method arms the acquisition system and waits for a user-specified time for a trigger. The second argument, a Boolean, specifies whether or not to force a trigger before returning if a trigger doesn't arrive within the allotted time period. The method also returns a Boolean value signifying whether or not a trigger arrived. See the reference section for more information on this useful method.

Another scenario where synchronization is necessary is between changing settings and reading results, even when no acquisition took place. For this the **WaitUntilIdle** method is used. This method is "blocking" and will not return control until the setup request has completed.

*Note that the **Acquire** method is equivalent to setting the Trigger Mode to "Single", then executing **WaitUntilIdle**.*

An example of **WaitUntilIdle** usage follows:

```
Sub Button1_Click()  
    ' Connect to the DSO  
    Set app = CreateObject("LeCroy.XStreamDSO")  
  
    ' Enable Standard Vertical parameters  
    app.Measure.MeasureMode = "StdVertical"  
  
    ' Wait for the change to take place for a max. of 5 seconds  
    app.WaitUntilIdle(5)  
  
    ' Read the value of measurement P1 (pkpk) and transfer into the spreadsheet  
    Cells(1, 2).Value = app.Measure.P1.Out.Result.Value  
  
    ' Enable Standard Horizontal parameters  
    app.Measure.MeasureMode = "StdHorizontal"  
  
    ' Wait for the change to take place for a max. of 5 seconds  
    app.WaitUntilIdle(5)  
  
    ' Read the value of measurement P1 (rise time) and transfer into the  
    spreadsheet  
    Cells(1, 3).Value = app.Measure.P1.Out.Result.Value  
End Sub
```

**NOTE:** In almost all remote control applications, it is **HIGHLY RECOMMENDED** that you **STOP** acquisitions before accessing result data. Most remote control problems are caused by failure to follow this practice.

## PART ONE: ABOUT AUTOMATION

---

### GOOD PRACTICES

- Using the **app.SetToDefaultSetup** action, restore the instrument to its default state before setting the controls required by an application. This eliminates any dependency on the previous configuration of the instrument. LeCroy strives to ensure that the default state of the instrument is constant from one software release to the next.
- Synchronization is an important concept that needs to be understood before you work with an X-Stream DSO via Automation. Attempting to read results while acquisitions are in progress could cause inconsistent results.
- Use the X-Stream Browser while developing Automation applications. This tool is guaranteed to show the up-to-date status of the Automation hierarchy since it retrieves it from a running instrument. It is also a very quick and easy way to exercise controls in real-time without your having to write a single line of code.
- When using a result object, verify that the status is valid to ensure that the acquisition and/or processing was valid.

### EXAMPLES

Following are fairly complete examples of automating an X-Stream DSO, including configuration, acquisition, and reading results. Examples are given both as Excel macros, and as Visual Basic Scripts, which can run without Excel being loaded on the instrument.

#### Example 1: Excel Macro to Perform FFT of C1

```
Sub Button1_Click()
    ' Connect to the DSO
    Set app = CreateObject("LeCroy.XStreamDSO")

    ' Restore the instrument to its default state
    app.SetToDefaultSetup

    ' Stop acquisitions during setup
    app.Acquisition.TriggerMode = "Stopped"

    ' Turn C2 off (default state leaves C1 and C2 On)
    app.Acquisition.C2.View = False

    ' Configure F1=FFT(C1), using a Blackman-Harris filter
    app.Math.F1.View = True
    app.Math.F1.Source1 = "C1"
    app.Math.F1.Operator1 = "FFT"
    app.Math.F1.Operator1Setup.Window = "BlackmanHarris"

    ' Take a single acquisition, force after 2 seconds if it doesn't trigger
    app.Acquisition.Acquire 2, True

    ' Read out the FFT
    ' Query the number of samples in F1 and store in cell "B1"
    numSamples = app.Math.F1.Out.Result.Samples
    Cells(1, 2).Value = numSamples

    ' Access the waveform data array, and fill the first column
    ' of the spreadsheet with it
    wave = app.Math.F1.Out.Result.DataArray
    For i = 0 To numSamples - 1
```

```
Cells(i + 1, 1).Value = wave(i)
Next
```

```
End Sub
```

## Example 2: VBScript Program to Perform FFT of C1 and Store Results in Text File

This example requires no additional software to be installed on the instrument, since it relies upon the built-in Visual Basic Script interpreter. The example is very similar to the previous Excel example, the most notable difference being the use of a standard system ActiveX control, “**Scripting.FileSystemObject**”, to enable the creation of files containing waveform data in ASCII format.

```
' VBScript example
' Configure the DSO to perform an FFT on Channel 1 and store
' the resulting data in a text file in ASCII format

' Connect to the DSO
Set app = CreateObject("LeCroy.XStreamDSO")

' Restore the instrument to its default state
app.SetToDefaultSetup

' Stop acquisitions during setup
app.Acquisition.TriggerMode = "Stopped"

' Turn C2 off (default state leaves C1 and C2 On)
app.Acquisition.C2.View = False

' Configure F1=FFT(C1), using a Blackman-Harris filter
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.Operator1 = "FFT"
app.Math.F1.Operator1Setup.Window = "BlackmanHarris"

' Take a single acquisition, force after 2 seconds if it doesn't trigger
app.Acquisition.Acquire 2, True

' Readout the FFT
numSamples = app.Math.F1.Out.Result.Samples

Set fso = CreateObject("Scripting.FileSystemObject")
Set MyFile= fso.CreateTextFile("c:\XStreamFFT.txt", True)

' Write the FFT power spectrum into the file, sample by sample
wave = app.Math.F1.Out.Result.DataArray
For i = 0 To numSamples - 1
    MyFile.WriteLine(wave(i))
Next

' Clean up
MyFile.Close
Set fso = Nothing
Set app = Nothing
```

## PART ONE: ABOUT AUTOMATION

---

### Example 3: Script to Measure the Rise Time of the Signal on C1 and Display It in a Popup Window

This example configures the DSO to measure the rise time of the signal on C2, take a single acquisition, and present the results in a popup dialog. The example requires no additional software to be installed on the instrument, since it relies on the built-in Visual Basic Script interpreter.

```
' VBScript example
' Configure the DSO to measure the rise time of the signal
' on Channel 1 and display it in a popup message box.

' Connect to the DSO
Set app = CreateObject("LeCroy.XStreamDSO")

' Restore the instrument to its default state
app.SetToDefaultSetup

' Stop acquisitions during setup
app.Acquisition.TriggerMode = "Stopped"

' Turn C2 off (default state leaves C1 and C2 On)
app.Acquisition.C2.View = False

' Configure Pl=rise(C1)
app.Measure.MeasureMode = "MyMeasure"
app.Measure.P1.View = True
app.Measure.P1.ParamEngine = "rise"

' Take a single acquisition, force after 2 seconds if it doesn't trigger
app.Acquisition.Acquire 2, True

' Present the rise time in a popup message box
MsgBox app.Measure.P1.Out.Result.Value & "s", vbOKOnly, "Rise time of C1"

' Clean up
Set app = Nothing
```

## EARLY AND LATE BINDING

The COM standard on which Automation is built supports two kinds of “binding” between client and server: early (static), and late (dynamic, dispatch). Static binding usually involves a type library and is used primarily by compiled languages such as C++. In this case, function entry points are resolved at compile time. Dynamic binding (also known as late binding) involves resolving method and property calls at run time, as opposed to compile time.

The Automation interfaces in X-Stream based DSOs use primarily the latter: Dynamic binding. From most programming languages (VB, VBScript, etc.) this is transparent. But when you are developing applications in C++, which doesn't provide late-binding natively, the use of a “helper” class is required. This is demonstrated below:

```
#include "stdafx.h"

#include "AtlBase.h"
CComModule _Module;
#include "AtlCom.h"

CComPtr<IDispatch> spDso;
CComDispatchDriver ddDso;    // dispatch ptr. to root of object model (app)

int main(int argc, char* argv[])
{
    printf("Hello X-Stream World!\n");

    ::CoInitialize(NULL);

    HRESULT hr = spDso.CoCreateInstance(L"LeCroy.XStreamDSO");
    if(SUCCEEDED(hr))
    {
        ddDso = spDso;

        // perform an Auto-Setup (app.AutoSetup)
        hr = ddDso.Invoke0(L"AutoSetup");

        // retrieve a Dispatch ptr. to the app.Display object
        CComVariant displayPtr;
        hr = ddDso.GetPropertyByName(L"Display", &displayPtr);
        CComDispatchDriver ddDisplay(displayPtr.pdispVal);

        // enter Dual-grid mode (app.Display.GridMode = "Dual")
        hr = ddDisplay.PutPropertyByName(L"GridMode", &CComVariant("Dual"));
    }

    return 0;
}
```

## PART ONE: ABOUT AUTOMATION

---

### VBS REMOTE COMMAND

For users who wish to harness the power of Automation control of an instrument, but are currently using “traditional” remote commands via GPIB or the network (using the VICP protocol), there is a solution. This is primarily of interest in controlling the advanced features of X-Stream DSOs, which are not supported by a traditional remote command.

X-Stream instruments, in addition to supporting LeCroy’s standard remote command set, also support a new command/query called **VBS[?]**. This command may be used in traditional remote control applications to access Automation controls.

This example shows two methods for setting the V/Div of Channel 1, the former using a traditional remote command, **VDIV**, and the latter using an Automation control via the new remote command, **VBS**. These two commands are equivalent:

```
C1:VDIV 0.5
```

```
VBS `app.Acquisition.C1.VerScale = 0.5`
```

In its query form the following are equivalent:

```
C1:VDIV?
```

```
VBS? `return = app.Acquisition.C1.VerScale`
```

A couple of points to note here are that the **app** variable is pre-defined and refers to the root of the Automation hierarchy. Also note that for the query form the **return =** is important, it indicates which value you wish to return to the caller.

The VBS[?] Command/Query is documented in more detail in the *Remote Control Manual* for X-Stream DSOs.

# X-STREAM DSO OBJECTS

The object hierarchy exposed by X-Stream based instruments is rooted at the Application object. This is the object returned when the **CreateObject("LeCroy.XStreamDSO")** method is executed in Visual Basic. All major instrument subsystems are available from this object, and many of these subsystems themselves are broken down further. Note that to simplify this figure only the first and last of the collections of Channels, Memories, Math, and Measurements are shown.

The reference section of this manual describes the controls presented by each of these objects.



**NOTE:** The root of the object hierarchies of some software options are not shown in this diagram.



# CONTROL REFERENCE

This section presents a reference guide for each object and control in the X-Stream object hierarchy.

## PART TWO: REFERENCE

**LeCroy.XStreamDSO.1****app**

This is the root of the automation hierarchy, all other nodes are accessed from this point.

<b>AddZoomTrace</b>	Action
<b>AutoSetup</b>	Action
<b>ClearSweeps</b>	Action
<b>DoPrint</b>	Action
<b>ExitWithoutConfirm</b>	Action
<b>FirmwareVersion</b>	String
<b>Height</b>	Property
<b>HideClock</b>	Bool
<b>InstrumentID</b>	String
<b>InstrumentModel</b>	String
<b>Left</b>	Property
<b>Minimize</b>	Action
<b>Quit()</b>	Method
<b>ResetPreferences</b>	Action
<b>SetToDefaultSetup</b>	Action
<b>Shutdown</b>	Action
<b>Sleep([in] double timeoutMilliseconds)</b>	Method
<b>Top</b>	Property
<b>TouchScreenEnable</b>	Bool
<b>WaitUntillIdle([in] double timeoutSeconds)</b>	Method
<b>Width</b>	Property
<b>Windowed</b>	Action
<b>WindowState</b>	Property

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Pop up a dialog containing the instrument model
MsgBox "Model is: " & app.InstrumentModel
```

**AddZoomTrace****Action**

## Description

Creates function traces, defined as zoom, for each visible channel trace.  
Equivalent to the front-panel zoom button.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Create zooms of all visible channel traces
app.AddZoomTrace
```

**CHAPTER TWO****Control Reference****AutoSetup****Action**

## Description

Starts an AutoSetup operation. When input channels are visible, AutoSetup operates only on those visible channels. If no channels are visible, all channels are affected by AutoSetup. When more than one channel is visible, the first visible channel in numerical order (that has a detectable signal applied to it) is automatically set up for edge triggering.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Start an Auto-Setup process.
app.AutoSetup
```

**ClearSweeps****Action**

## Description

Clears all accumulated sweeps for all subsystems. These include Channel Pre-Processing, Math, Measure, and Display Persistence. Subsystem-specific clear sweeps controls are also available. For details, please refer to the ClearSweeps control for each subsystem.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Clear all accumulated sweeps for all subsystems.
app.ClearSweeps
```

**DoPrint****Action**

## Description

Executes a printout of the currently displayed screen image. The destination printer or file, and various other options, are defined in the Hardcopy subsystem (app.Hardcopy).

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Start a print of the screen to the pre-determined destination.
app.DoPrint
```

## PART TWO: REFERENCE

---

### ExitWithoutConfirm

**Action**

#### Description

Causes the instrument application to exit without prompting for a confirmation. Any acquisition in progress will be canceled.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Shut down the instrument application.
app.ExitWithoutConfirm
```

### FirmwareVersion

**String**

Range: Any number of characters

#### Description

Queries the firmware version of the instrument. The response takes the form: "1.0.0 (build 12345)"

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Query the firmware version number of the instrument.
MsgBox "Firmware Version is: " + app.FirmwareVersion
```

### Height

**Property**

#### Description

Sets/Queries the height in pixels of the instrument display on the PC screen.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the height of the instrument window to 400 pixels.
app.Height = 400
```

### HideClock

**Bool**

#### Description

Hides/Shows the clock that resides in the lower-right corner of the display of the instrument.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Hide the clock for 3 seconds.
```

**CHAPTER TWO****Control Reference**

```

app.HideClock = True
app.Sleep(3000)
app.HideClock = False

```

**InstrumentID****String**

Range: Any number of characters

## Description

Reads the complete ID of the instrument in the format: "LECROY,WM8500,WM000001,0.0.0", which includes the maker, instrument model number, serial number, and version number.

## Example

```

' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Present the ID of the instrument.
MsgBox app.InstrumentID

```

**InstrumentModel****String**

Range: Any number of characters

## Description

Queries the model number of the instrument.

## Example

```

' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Present the model number of the instrument.
MsgBox app.InstrumentModel

```

**Left****Property**

## Description

Sets/Queries the position in pixels of the left edge of the instrument display on the PC screen. The position is measured from the left edge of the screen to the left edge of the instrument window.

## Example

```

' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the position of the left edge of the instrument window to
100 pixels.
app.Left = 100

```

## PART TWO: REFERENCE

---

### Minimize

### Action

#### Description

Minimizes the instrument window to reveal the underlying desktop. It will display a small window in the bottom-right corner of the display, which, when clicked, will restore the window to full-screen mode. To programmatically restore the window, refer to the `app.WindowState` control.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Minimize the instrument display.
app.Minimize
```

### Quit()

### Method

#### Description

Closes the instrument application. The instrument will prompt you with an "Are you sure?" dialog before closing down. Until you respond to the dialog, control via Automation will be blocked.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Quit the instrument application with a confirmation prompt.
app.Quit
```

### ResetPreferences

### Action

#### Description

Resets all scope preferences to their default states. The set includes the current remote communications port, the color palette settings, etc., but does not include main DSO controls such as V/Div, T/Div, etc. These main instrument controls can be reset using the `SetToDefaultSetup` control.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Reset all instrument preferences.
app.ResetPreferences
```

**SetToDefaultSetup****Action**

## Description

Restores the instrument setup to its default state. However, certain settings will not be restored to the default state. These are the user preferences such as current remote communications port and color settings, which can be reset, if required, using the ResetPreferences action.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Restore the instrument to its default state.
app.SetToDefaultSetup
```

**Shutdown****Action**

## Description

Shuts down the instrument. It will prompt you with an "Are you sure?" dialog before shutting down. Until you respond to the dialog, control via Automation will be blocked.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Shut down the instrument with a confirmation prompt.
app.Shutdown
```

**Sleep([in] double timeoutMilliseconds)****Method**

## Description

Causes the main execution thread of the instrument application to sleep for the specified time period, defined in milliseconds.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

MsgBox "Sleeping for 10 seconds..."
app.Sleep(10000)
MsgBox "Sleep finished"
```

## PART TWO: REFERENCE

---

### Top

### *Property*

#### Description

Sets/Queries the position in pixels of the top edge of the instrument display on the PC screen. The position is measured from the top of the screen to the top of the instrument window.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the position of the top edge of the instrument window to
100 pixels.
app.Top = 100
```

### TouchScreenEnable

### *Bool*

#### Description

Sets/Queries the state of the touch-screen enable control. This = the front-panel Touch Screen button.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Disable touch-screen if it is enabled.
if app.TouchScreenEnable = True then
    app.TouchScreenEnable = False
End if
```

### WaitUntilIdle([in] double timeoutSeconds)

### *Method*

#### Description

Waits until either the application is idle or the specified timeout (in seconds) expires. This evaluates to True if the application completes before the timeout expires, and to False if a timeout occurs. When the trigger mode is Auto or Run, the application is never Idle. In this case the call to WaitUntilIdle returns after the next acquisition and any configured processing.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Wait with a timeout of five seconds.
app.WaitUntilIdle(5)
```

**Width****Property**

## Description

Sets/Queries the width in pixels of the instrument display on the PC screen.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the width of the instrument window to 800 pixels.
app.Width = 800
```

**Windowed****Action**

## Description

Places the instrument application in windowed mode (as opposed to full-screen mode). Places the application in the upper part of the display screen with a sizable border.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the instrument display into the windowed mode.
app.Windowed
```

**WindowState****Property**

## Description

Sets/Queries the state of the PC window used by the instrument display.

- 0 windowed
- 1 full screen
- 2 minimized

Trying to set values greater than 2 or less than 0 will result in the value

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the instrument window state to windowed.
app.WindowState = 0
```

## PART TWO: REFERENCE

## ACQUISITION

*app.Acquisition*

This group of variables controls the input channels (C1, C2, C3, C), the timebase, the trigger, and the Aux Output.

Names of the form `app.Acquisition.Channels.xxxx` are aliases for simpler names, which are described in this section of the manual. Examples of alias pairs are as follows:

```
app.Acquisition.Channels("Cx") = app.Acquisition.Cx
app.Acquisition.Channels(1) = app.Acquisition.C1
app.Acquisition.Channels("Cx").Out.Result = app.Acquisition.Cx.Out.Result
```

These longer names may be more suitable than the shorter ones for certain programming structures.

**Acquisition**

<b>Acquire([in] double timeoutSeconds, [in] long bForceTriggerOnTimeout)</b>	Method
<b>Calibrate</b>	Action
<b>ClearSweeps</b>	Action
<b>TriggerMode</b>	Enum

**Acquire([in] double timeoutSeconds, [in] long****Method**

## Description

Action/Query. Takes a single acquisition. The first of the two arguments specifies a timeout; the second, which is optional, specifies whether or not to force a trigger when the timeout occurs.

Evaluates to True if a trigger occurred, or False if a timeout occurred.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Start an acquisition, wait for up to 5 seconds for a trigger
' event, force a software trigger if a hardware trigger is not
' detected before the 5 second timeout expires.
triggerDetected = app.Acquisition.Acquire(5, true)
```

**Calibrate****Action**

## Description

Initiates a full calibration of the acquisition system of the instrument.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Start a calibration.
app.Acquisition.Calibrate
```

**ClearSweeps****Action**

## Description

Resets any accumulated average data or persistence data for channel waveforms (C1– C4). Valid only when one or more channels have waveform averaging or persistence enabled in their preprocessing settings. An average can be reset on an individual basis using app.Acquisition.Cx.ClearSweeps control.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Clear accumulated sweeps for channels C1...C4
app.Acquisition.ClearSweeps

' Clear accumulated sweeps for only C1
app.Acquisition.C1.ClearSweeps
```

**TriggerMode****Enum**

## Description

Sets/Queries the trigger mode, using values from the following list:

<b>Auto</b>	After a timeout, if a real hardware trigger is not received, the instrument will force a trigger so that there are frequent automatic updates.
<b>Normal</b>	Accepts triggers as rapidly as the system permits, but will wait indefinitely for a trigger, without updating data.
<b>Single</b>	Arms the acquisition system to acquire once, and does not rearm automatically afterward. Once a trigger is received and the data is processed, the instrument enters the "Stopped" state.
<b>Stop</b>	Finishes the current acquisition and does not re-arm.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Place the instrument in stopped mode and take one acquisition.
app.Acquisition.TriggerMode = "Stopped"
app.Acquisition.Acquire(5)
```

**PART TWO: REFERENCE**

---

Values

<b>Auto</b>	Auto-trigger
<b>Normal</b>	Normal Trigger
<b>Single</b>	Single Trigger
<b>Stopped</b>	No trigger possible, Stopped

**AUXOUTPUT***app.Acquisition.AuxOutput*

Controls for the Auxiliary output BNC can be programmed as a simple square-wave signal source, or as a pulse that is asserted when various events occur, including Trigger Enabled, Trigger Out, and Pass/Fail.

<b>Amplitude</b>	Double
<b>AuxInCoupling</b>	Enum
<b>Frequency</b>	DoubleLockstep
<b>Mode</b>	Enum
<b>PulseWidth</b>	Double
<b>SetToTTL</b>	Bool

**Example**

```
' Microsoft Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Setup the Auxiliary output to be a square wave with an amplitude

' of 500mV a frequency of 5kHz
app.Acquisition.AuxOutput.Mode = "Square"
app.Acquisition.AuxOutput.Amplitude = 0.5
```

**Amplitude****Double**

Range: From 0.005 to 1, step 0.001

**Description**

Sets/Queries the amplitude of the signal on the AUX OUT connector. This is the amplitude of the signal into a 1 Mohm load. Into 50 ohms the output voltage will be halved (since the source impedance is nominally 50 ohms). Units are Volts.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the amplitude of the signal from the AUX OUT connector
' to 0.6 V into 1 Mohm, or 0.3 V into 50 ohms.
app.Acquisition.AuxOutput.Amplitude = 0.6
```

## PART TWO: REFERENCE

---

### AuxInCoupling

*Enum*

#### Description

Sets the input coupling for the Auxiliary input path.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the coupling of the Auxiliary socket, when used as an
input, to ground.
' In this condition, no input signal reaches the instrument.
app.Acquisition.AuxOutput.AuxInCoupling = "GND"
```

#### Values

<b>DC50</b>	DC, 50 ohms coupling
<b>GND</b>	Grounded

### Frequency

*DoubleLock step*

Range: From 5 to 5e+006 step 10, locked to 1 2.5 5

#### Description

Sets/Queries the auxiliary output frequency of the square wave. Units are Hertz. WaveMaster models (and derivatives) have a limit of 5 MHz. WavePro 7000 models (and derivatives) have a limit of 1 MHz.

This control only has effect when the AuxOutput mode is "Square".

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the frequency of the signal from the AUX OUT
connector to 1 MHz.
app.Acquisition.AuxOutput.Frequency = 1e6
```

### Mode

*Enum*

#### Description

Sets/Queries the output mode of the AUX OUT connector.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the output of the AUX OUT connector to output
' a pulse on a pre-determined Pass-Fail decision.
app.Acquisition.AuxOutput.Mode = "PassFail"
```

Values

<b>DCLevel</b>	Emit a DC level
<b>Off</b>	Output Disabled
<b>PassFail</b>	Pulse-out controlled by Pass/Fail system
<b>Square</b>	Square-wave signal generator
<b>TriggerEnabled</b>	Pulse-out when trigger is enabled
<b>TriggerOut</b>	Pulse-out when trigger occurs

**PulseWidth****Double**

Range: From 0.001 to 0.5, step 0.001

Description

This control has effect only if the Aux Output is in pulse mode (e.g., pass/fail, trigger out). It has no effect for modes that do not produce a pulse.

Sets the duration of the output pulse from the AUX OUT connector. Units are seconds.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the duration of the pulse from AUX OUT
' to 15 ms.
app.Acquisition.AuxOutput.PulseWidth = 15e-3
```

**SetToTTL****Bool**

Description

Overrides the Amplitude setting when true. This control Sets the Auxiliary output to produce TTL levels.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the levels of the signal from the AUX OUT
' connector to produce TTL levels.
app.Acquisition.AuxOutput.SetToTTL = True
```

## PART TWO: REFERENCE

**CHANNELS***app.Acquisition.Channels*

This group of variables controls the acquisition channels C1, C2, C3, and C4.

Names of the form `app.Acquisition.Channels.xxxx` are aliases for simpler names, which are described in the section of the manual devoted to `app.Acquisition`. Examples of alias pairs are as follows:

```
app.Acquisition.Channels("Cx") = app.Acquisition.Cx
app.Acquisition.Channels(1) = app.Acquisition.C1
app.Acquisition.Channels("Cx").Out.Result = app.Acquisition.Cx.Out.Result
```

**Channels**

## Example

```
Set app = CreateObject("LeCroy.XStreamDSO")
For X = 1 To 4
    app.Acquisition.Channels(X).VerScale = 0.2
Next
```

**CX***app.Acquisition.Cx*

This group of variables controls the input channels C1, C2, C3, and C4.

Names of the form `app.Acquisition.Channels.xxxx` are aliases for simpler names, which are described in the section of the manual devoted to `app.Acquisition`. Examples of alias pairs are as follows:

```
app.Acquisition.Channels("Cx") = app.Acquisition.Cx
app.Acquisition.Channels("Cx").Out.Result = app.Acquisition.Cx.Out.Result
```

<b>AverageSweeps</b>	Integer
<b>AxisXRotation</b>	Integer
<b>AxisYRotation</b>	Integer
<b>BandwidthLimit</b>	Enum
<b>ClearSweeps</b>	Action
<b>Coupling</b>	Enum
<b>Deskew</b>	Double
<b>InterpolateType</b>	Enum
<b>Invert</b>	Bool
<b>LabelsPosition</b>	String
<b>LabelText</b>	String
<b>Persist3DQuality</b>	Enum
<b>Persisted</b>	Bool
<b>Persistence3d</b>	Bool
<b>PersistenceMonoChrome</b>	Bool
<b>PersistenceSaturation</b>	Integer
<b>PersistenceTime</b>	Enum
<b>ProbeAttenuation</b>	Double
<b>ShowLastTrace</b>	Bool
<b>UseDotJoin</b>	Bool
<b>UseGrid</b>	String
<b>VerOffset</b>	Double
<b>VerScale</b>	DoubleLockstep

<b>VerScaleVariable</b>	Bool
<b>View</b>	Bool
<b>ViewLabels</b>	Bool

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Setup Channel C1
app.Acquisition.C1.VerScale = 0.5
app.Acquisition.C1.VerOffset = 0.0
app.Acquisition.C1.Coupling = "DC50"

' Setup Channel C2
app.Acquisition.C2.VerScale = 0.1
app.Acquisition.C2.VerOffset = 0.2
app.Acquisition.C2.Coupling = "DC50"
```

**AverageSweeps***Integer*

Range: From 1 to 1000000, step 1

**Description**

Sets/Queries the number of averaging sweeps for input channel Cx. This is distinct from the math function app.Math.Fx. If the number of sweeps is 1 (the default value), the data will not be averaged.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the number of sweeps for channel C1 to 25.
app.Acquisition.C1.AverageSweeps = 25
```

**AxisXRotation***Integer*

Range: From -90 to 90, step 1

**Description**

Sets/Queries the state of the X-axis rotation control, used only in 3-D persistence modes to control the viewing position.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the rotation about the X-axis to 35 degrees for trace C3.
app.Acquisition.C3.AxisXRotation = 35
```

## PART TWO: REFERENCE

---

### AxisYRotation

*Integer*

Range: From -90 to 90, step 1

#### Description

Sets/Queries the state of the Y-axis rotation control, used only in 3-D persistence modes to control the viewing position.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the rotation about the Y-axis to 25 degrees for trace C3.
app.Acquisition.C3.AxisYRotation = 25
```

### BandwidthLimit

*Enum*

#### Description

Sets/Queries the bandwidth limit for input channel Cx, in Hz. This control is an enum, and therefore requires a string value, and not a scalar value. Bandwidth limit choices vary between DSO models.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the bandwidth limit for C2 to 20 MHz.
app.Acquisition.C2.BandwidthLimit = "20MHz"
```

#### Values

1GHz  
200MHz  
20MHz  
3GHz  
4GHz  
Full

### ClearSweeps

*Action*

#### Description

Clears all accumulated average data and persistence data for this channel. See **app.Acquisition.ClearSweeps** for a control that clears accumulated data for channels 1 to 4, or **app.ClearSweeps** for a control that clears accumulated data for all subsystems (including Math/Measure/Display, etc.)

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Reset channel C1
app.Acquisition.C1.ClearSweeps
```

## CHAPTER TWO

## Control Reference

```
' Reset channels C1..C4
app.Acquisition.ClearSweeps
```

**Coupling****Enum**

## Description

Sets/Queries the input coupling of input channel Cx. Coupling choices vary between instrument models. WavePro 7000 instruments, for example, support AC1M and DC1M modes in addition to DC50 and GND.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the input coupling for channel C2
CoupleC2 = app.Acquisition.C2.Coupling

' Set the coupling to DC, 50 ohms
app.Acquisition.C2.Coupling = "DC50"
```

## Values

```
DC50
Gnd
```

**Deskew****Double**

Range: From -0.1 to 0.1, step 1e-012

## Description

Sets/Queries the deskew of input channel Cx to produce a required alignment with another trace.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the deskew of channel C2 to 3.0 ns
app.Acquisition.C2.Deskew = 3.0e-9
```

**InterpolateType****Enum**

## Description

Sets/Queries the type of interpolation used for input channel Cx. Because Sinx/x interpolation increases the size of the trace by a factor of 10, beware when using this option with long records.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the interpolation for channel C3 to (sin x)/x
app.Acquisition.C3.InterpolateType = "Sinxx"
```

## PART TWO: REFERENCE

---

Values

<b>Linear</b>	Linear interpolation
<b>Sinxx</b>	Sinx/x interpolation

### Invert

**Bool**

Description

Sets/Queries whether input channel Cx is inverted.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set channel C2 to be inverted.
app.Acquisition.C2.Invert = True
```

### LabelsPosition

**String**

Range: Any number of characters

Description

Sets/Queries the horizontal position of the label attached to acquisition trace Cx. The unit of measurement is the unit of the horizontal scale. The measurement is made from the trigger point. This control is a string, not a numeric value. This allows multiple labels to be positioned, as shown in the example below.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Add a couple of labels to trace C1, one at 0ns, and one at 55ns

app.SetToDefaultSetup
app.Acquisition.C1.ViewLabels = True
app.Acquisition.C1.LabelsPosition = "0.0,55e-9"
```

### LabelsText

**String**

Range: Any number of characters

Description

Sets/Queries the text that appears in labels attached to acquisition trace Cx. Multiple labels can be specified by using comma as a delimiter.

**Persist3DQuality****Enum**

## Description

Sets/Queries the state of the 3-D Persistence quality control, which controls the way that the persistence trace is rendered.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set persistence 3-D to shaded for trace C2.

app.Acquisition.C2.Persist3DQuality = "Shaded"
```

## Values

Shaded  
Solid  
WireFrame

**Persisted****Bool**

## Description

Sets/Queries the persisted state of the channel waveform. If the Display.LockPersistence control is set to "AllLocked" then the persisted state of all displayed waveforms will be the same. If the Display.LockPersistence control is set to "PerTrace" then the persisted state of each waveform can be independently controlled.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set persistence on for trace C1
app.Display.LockPersistence = "PerTrace"
app.Acquisition.C1.Persisted = True
```

**Persistence3d****Bool**

## Description

Sets/Queries the 3-D persistence state. When True, the persistence display for this channel will be displayed as a three-dimensional surface map.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set persistence plot as 3-D for trace C1
app.Acquisition.C1.Persistence3D = True
```

## PART TWO: REFERENCE

---

### PersistenceMonoChrome

**Bool**

#### Description

Sets/Queries the monochrome persistence state. When True, the persistence display for this channel will be monochromatic, whether 2-D or 3-D.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set persistence monochrome on for trace C4.
app.Acquisition.C4.PersistenceMonoChrome = True
```

### PersistenceSaturation

**Integer**

Range: From 1 to 100, step 1

#### Description

Sets/Queries the saturation threshold for persisted waveforms. All information at this level or above will be recorded with the same color or intensity.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the persistence saturation level for trace C1.
app.Acquisition.C1.PersistenceSaturation = 60
```

### PersistenceTime

**Enum**

#### Description

Sets/Queries the state of the Persistence Time control. Controls the persistence decay time for this trace.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the persistence time for the persistence trace of channel
C1 to 10 seconds.
app.Acquisition.C1.PersistenceTime = "10s"
```

#### Values

0.5s	5s
10s	Infinite
1s	2s
20s	

**ProbeAttenuation****Double**

Range: From 1e-006 to 10000, step 1e-006

## Description

Sets/Queries the probe attenuation. The probe attenuation is the factor by which the signal is made smaller, for example, 10 means that the probe divides by 10, and is referred to as a ÷10 probe. Certain passive probes can be marked as "x10", even though they actually divide the input signal by a factor of 10.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the probe attenuation for channel C1 to 100
app.Acquisition.C1.ProbeAttenuation = 100
```

**ShowLastTrace****Bool**

## Description

Sets/Queries the state of the Show Last Trace control. If True, when this trace is displayed in persistence mode, the last acquired waveform will be superimposed on the accumulating persistence map.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Makes the last acquired trace invisible for the
' persistence trace of channel C1.
app.Acquisition.C1.ShowLastTrace = False
```

**UseDotJoin****Bool**

## Description

Sets/Queries the DotJoin state of the channel. If True then straight line segments will be drawn between sample points. If false then only the sample points will be shown. See **Display.TraceStyle** for a control that can change the setting for all displayed traces simultaneously.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Turn dot joining off for trace C1.
app.Acquisition.C1.UseDotJoin = False
```

## PART TWO: REFERENCE

---

### UseGrid

*String*

Range: Any number of characters

#### Description

Sets/Queries the graticule on which the trace is displayed. Typical values include:

<b>YT1 to YT8</b>	One of the YT graticules used in Single, Dual, Quad, and Octal display modes
<b>NotOnGrid</b>	Not displayed on any graticule

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Switch to dual grid mode, place C1 on the lower graticule
' and C2 on the upper graticule.
app.Display.GridMode = "Dual"
app.Acquisition.C1.UseGrid = "YT2"
app.Acquisition.C2.UseGrid = "YT1"
```

### VerOffset

*Double*

Range: From -0.75 to 0.75, step 0.001

#### Description

Sets/Queries the vertical offset of input channel Cx. The setting resolution in volts lies in the range 0.25% to 0.5%, depending on the numerical value. The available offset range depends on the current V/Div setting, and also the instrument model.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the vertical offset for C1 to 10 mV.
app.Acquisition.C1.VerOffset = 0.01
```

### VerScale

*DoubleLock step*

Range: From 0.002 to 1, step 0.0005, locked to 1 2 5

#### Description

Sets/Queries the vertical scale (in Volts/Division) of an input channel. When variable gain (VerScaleVariable control) is disabled, the control will clip values to a 1-2-5 sequence. When it is enabled, the setting resolution lies in the range 1% to 2%, depending upon the numerical value.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

## CHAPTER TWO

## Control Reference

```
' Set C1 to a scale of 250mV/Div in Variable Scale mode
app.Acquisition.C1.VerScaleVariable = True
app.Acquisition.C1.VerScale = 0.25
```

**VerScaleVariable****Bool**

## Description

Sets/Queries the state of the variable vertical scale control for channel Cx. When the variable scale is enabled, the setting resolution lies in the range 1% to 2%, depending on the numerical value. If a knowledge of the exact value is important, the value should be read back after a setting has been made.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the variable vertical scale for C1 to On.
app.Acquisition.C1.VerScaleVariable = True
```

**View****Bool**

## Description

Sets/Queries the channel's "Viewed" state. When True, the channel waveform is displayed on one of the display gratitudes. Even when a channel is not visible, it can be used as a source for Math, Measure, etc.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Make channel C3 visible.
app.Acquisition.C3.View = True
```

**ViewLabels****Bool**

## Description

Sets/Queries whether the user-defined labels for trace Cx is visible. See Also: **LabelsPosition** and **LabelsText** controls.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Show the user-defined label for trace C2.
app.Acquisition.C2.ViewLabels = True
```

## PART TWO: REFERENCE

### RESULT

### app.Acquisition.Cx.Out.Result

Properties of the type xxxx.Out.Result.yyyy are those of the last completed acquisition. They are not affected if other controls are changed after that acquisition was completed. This distinction between "Out.Result" properties and other controls is most important when the trigger mode is Single or Stopped. You should treat "Out.Result" properties as read-only.

Several of these properties mention the "frame" ? the term used to describe the visible portion of the trace, which is generally smaller than the acquired waveform. For example, the frame could be used to display a 500 pt. window onto a 1 Mpt. trace; or, vertically it could be used to show the "center" 10 mV of a result.

<b>DataArray</b>	Property
<b>FirstEventTime</b>	Property
<b>HorizontalFrameStart</b>	Property
<b>HorizontalFrameStop</b>	Property
<b>HorizontalOffset</b>	Property
<b>HorizontalPerStep</b>	Property
<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>IndexOfFirstSampleInFrame</b>	Property
<b>LastEventTime</b>	Property
<b>NumFrameDimensions</b>	Property
<b>NumSamplesInFrame</b>	Property
<b>Samples</b>	Property
<b>Status</b>	Property
<b>StatusDescription</b>	Property
<b>Sweeps</b>	Property
<b>VerticalFrameStart</b>	Property
<b>VerticalFrameStop</b>	Property
<b>VerticalMaxPossible</b>	Property
<b>VerticalMinPossible</b>	Property
<b>VerticalOffset</b>	Property
<b>VerticalPerStep</b>	Property
<b>VerticalResolution</b>	Property
<b>VerticalUnits</b>	Property

#### DataArray

#### Property

##### Description

This is the array of data that can be read out to represent the input waveform. The data will have 16-bit resolution ? VerticalPerStep, though the physical resolution will usually be less. See **VerticalResolution**. The optional boolean argument can be used to determine whether 16-bit integer data, or floating-point data is returned. True indicates that floating-point values are required; False indicates that integer values are required.

##### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Copy the output data array (floats) for trace C3 into an array.
```

## CHAPTER TWO

## Control Reference

```
Dim C3dataArray
C3dataArray = app.Acquisition.C3.Out.Result.DataArray(True)

' Emit the first two data values into a popup message box
```

**FirstEventTime****Property**

## Description

Queries the absolute trigger time of the acquisition, or that of the first sequence in a segmented acquisition. Times are returned encoded as a currency value (VT\_CY) within a variant, which allows the use of the full 64-bit resolution of the timestamp value. Values are referenced to 1 Jan 2000, with 1 ns resolution. VT\_CY values are stored as 64-bit (8 byte) two's complement integers, scaled by 10,000 to give a fixed-point number with 15 digits to the left of the decimal point, and 4 digits to the right.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the time of the first event.
EventFirst = app.Acquisition.C1.Out.Result.FirstEventTime
MsgBox EventFirst
```

**HorizontalFrameStart****Property**

## Description

Reads the time, in seconds, that corresponds to the left edge of the graticule, relative to the trigger instant.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the horizontal frame start for trace C1
HorStart = app.Acquisition.C1.Out.Result.HorizontalFrameStart
MsgBox HorStart
```

**HorizontalFrameStop****Property**

## Description

Reads the time, in seconds, that corresponds to the right-hand edge of the graticule, relative to the trigger instant.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the horizontal frame stop for trace C1
HorStop = app.Acquisition.C1.Out.Result.HorizontalFrameStop
```

## PART TWO: REFERENCE

---

MsgBox HorStop

### HorizontalOffset

*Property*

#### Description

Reads/Queries the time between the trigger point and the left edge of the screen. If the trigger point is off the left edge of the screen, the result is positive. If the trigger point is on the screen, the result is negative. The value is very close to: `app.Acquisition.Horizontal.HorOffsetOrigin + app.Acquisition.Horizontal.HorOffset`, when both are expressed in seconds. A small discrepancy occurs because the trigger point is not synchronous with the sampling clock, so variations occur with a range of one sample period.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the value of the horizontal offset for trace C2
HorOffsetC2 = app.Acquisition.C2.Out.Result.HorizontalOffset
MsgBox HorOffsetC2
```

### HorizontalPerStep

*Property*

#### Description

Reads the time, in units of seconds, between successive sampling instants.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the value of the horizontal step size for trace C3
HorStepC3 = app.Acquisition.C3.Out.Result.HorizontalPerStep
MsgBox HorStepC3
```

### HorizontalResolution

*Property*

#### Description

Reads the resolution of the readout of horizontal values. It is not directly related to the sample period.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the readout resolution of the horizontal axis of trace C4
C4HRes = app.Acquisition.C4.Out.Result.HorizontalResolution
MsgBox C4HRes
```

## CHAPTER TWO

## Control Reference

**HorizontalUnits***Property*

## Description

Reads the unit in which the horizontal displacements are specified for trace Cx.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the units of the horizontal axis for trace C2.
C2HorUnits = app.Acquisition.C2.Out.Result.HorizontalUnits
MsgBox C2HorUnits
```

**IndexOfFirstSampleInFrame***Property*

## Description

Reads the index number of the first sample that appears in the frame.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the index of the first sample in the frame for trace C2.
FrameSampleC2 =
app.Acquisition.C2.Out.Result.IndexOfFirstSampleInFrame
MsgBox FrameSampleC2
```

**LastEventTime***Property*

## Description

Queries the time of the last contributing event in a set. Useful only when the result includes data produced by a sequence acquisition, or a cumulative operation such as averaging. See description for **FirstEventTime** for encoding details.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the time of the last event.
EventLast = app.Acquisition.C1.Out.Result.LastEventTime
MsgBox EventLast
```

## PART TWO: REFERENCE

---

### NumFrameDimensions

*Property*

#### Description

Reads the dimensionality of the trace Cx: 2 for a Y-T plot, 3 for an X-Y plot.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the dimensionality of trace C3.
DimC3 = app.Acquisition.C3.Out.Result.NumFrameDimensions
MsgBox DimC3
```

### NumSamplesInFrame

*Property*

#### Description

Reads the nominal number of samples in the displayed frame. In the case of sequence mode, the frame refers to one segment, not the whole graticule.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the number of samples in the frame for trace C1
FrameSamplesC1 = app.Acquisition.C1.Out.Result.NumSamplesInFrame
MsgBox FrameSamplesC1
```

### Samples

*Property*

#### Description

Reads the number of samples in a record, as read out. It will usually be two more than NumSampleInFrame (but may be much larger) to allow for the two samples that are just to the left and right of the displayed graticule. For a trace acquired in sequence mode, "frame" refers to one segment, not the whole graticule.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the number of samples in trace C1.
C1Samples = app.Acquisition.C1.Out.Result.Samples
MsgBox C1Samples
```

**Status****Property**

## Description

Queries the status of the waveform result. Status is a 64-bit bitfield, encoded in a VARIANT of VT\_CY (currency) type, with the meaning associated with each bit described earlier in Chapter 1. Status should be read twice, once before reading the result, and once again after reading the result. This is due to the streaming nature of processing in the X-stream DSO software.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

stat = app.Acquisition.C1.Out.Result.Status
MsgBox stat
```

**StatusDescription****Property**

## Description

Returns a textual equivalent of the status returned in the "Status" bitfield.

**Sweeps****Property**

## Description

Reads the number of trigger events (sweeps) that contributed to a cumulative result. Useful only for sequence acquisitions. Cumulative processing, such as Averaging. Accumulation, can be reset using the ClearSweeps method.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the number of accumulated sweeps for trace C1
' and display in a popup
numSweeps = app.Acquisition.C1.Out.Result.Sweeps
MsgBox numSweeps
```

**VerticalFrameStart****Property**

## Description

Reads the amplitude that corresponds to the bottom of the displayed frame.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the vertical frame start for trace C1.
VerStart = app.Acquisition.C1.Out.Result.VerticalFrameStart
MsgBox VerStart
```

## PART TWO: REFERENCE

---

### VerticalFrameStop

*Property*

#### Description

Reads the amplitude that corresponds to the top of the displayed frame.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the vertical frame stop for trace C1.
VerStop = app.Acquisition.C1.Out.Result.VerticalFrameStop
MsgBox VerStop
```

### VerticalMaxPossible

*Property*

#### Description

Reads the highest value that an actual array element can have. It will be a little less than VerticalFrameStop.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the vertical maximum possible value for trace C1.
VerMax = app.Acquisition.C1.Out.Result.VerticalMaxPossible
MsgBox VerMax
```

### VerticalMinPossible

*Property*

#### Description

Reads the lowest value that an actual array element can have. It will be a little greater than VerticalFrameStart.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the vertical minimum possible value for trace C1.
VerMin = app.Acquisition.C1.Out.Result.VerticalMinPossible
MsgBox VerMin
```

### VerticalOffset

*Property*

#### Description

Reads the difference in potential between ground and the center of the screen. With an offset of +50 mV, the center of the screen represents -50 mV; with an offset of -21 mV, the center represents +21 mV.

**CHAPTER TWO****Control Reference****Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the vertical offset for trace C1
VerOffset = app.Acquisition.C1.Out.Result.VerticalOffset
MsgBox VerOffset
```

**VerticalPerStep****Property****Description**

Reads the smallest step in the numerical values that can be read out, whether or not the step has physical meaning. For the basic 8-bit ADC, with values returned as 16-bit short values, the step is 1/65536 of the vertical range.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the vertical step size for trace C3.
VerStep = app.Acquisition.C3.Out.Result.VerticalPerStep
MsgBox VerStep
```

**VerticalResolution****Property****Description**

Reads the vertical resolution, which is the actual smallest difference that can be practically resolved. For an 8-bit ADC it is 1/256 of the height of the vertical range VR. But if 16 averages are set, the resolution is improved by a factor of 4, and it becomes 1/1024 of VR, and for 100 sweeps it becomes 1/2560.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the readout resolution of the vertical axis of trace C1
VRes = app.Acquisition.C1.Out.Result.VerticalResolution
MsgBox VRes
```

**VerticalUnits****Property****Description**

Reads the unit in which the vertical displacements are specified.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

## PART TWO: REFERENCE

---

```
' Read the units of the vertical axis of trace C1  
VertUnits = app.Acquisition.C1.Out.Result.VerticalUnits  
MsgBox VertUnits
```

**HORIZONTAL***app.Acquisition.Horizontal*

This group of variables controls the timebase, the sampling, and the trigger delay.

**Horizontal**

<b>AcquisitionDuration</b>	Double
<b>ActiveChannels</b>	Enum
<b>HorOffset</b>	Double
<b>HorOffsetControl</b>	Enum
<b>HorOffsetOrigin</b>	Double
<b>HorScale</b>	DoubleLockstep
<b>HorUnits</b>	String
<b>MaxSamples</b>	DoubleLockstep
<b>NumPoints</b>	Integer
<b>NumSegments</b>	Integer
<b>ReferenceClock</b>	Enum
<b>SampleClock</b>	Enum
<b>SampleMode</b>	Enum
<b>SampleRate</b>	DoubleLockstep
<b>SamplingRate</b>	Double
<b>SequenceTimeout</b>	Double
<b>SequenceTimeoutEnable</b>	Bool
<b>SmartMemory</b>	Enum
<b>TimePerPoint</b>	Double
<b>ZeroDelay</b>	Action

**AcquisitionDuration***Double*

Range: From 1e-012 to 1e+012, step 1e-015

## Description

Queries the duration of the last completed acquisition. The result may depend on the spacing of the triggers in sequence mode, and it may depend on the number of averages when a channel is in averaging mode.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Obtain the duration of the last completed acquisition.
AcqDuration = app.Acquisition.Horizontal.AcquisitionDuration
MsgBox AcqDuration
```

## PART TWO: REFERENCE

---

### ActiveChannels

*Enum*

#### Description

Sets/Queries the number of active DSO input channels. This is a string value, with allowed values "4", "2", and "Auto"; and 0, 1, and 2. Beware of using 2 as a numerical value for 2 channels: you will get Auto mode instead.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the instrument to use two channels.
app.Acquisition.Horizontal.ActiveChannels = "2"
```

#### Values

<b>2</b>	Use 2 channels, with increased sample rate
<b>4</b>	Use all channels
<b>Auto</b>	Maximize sample rate based upon the # displayed channels

### HorOffset

*Double*

Range: From -0.0005 to 5e-007, step 1e-009

#### Description

Sets/Queries the horizontal position of the trigger time, relative to the origin set by HorOffsetOrigin, in seconds. Positive to the right, negative to the left. The setting resolution is about 1% to 2%.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the horizontal trigger offset to 200 ns.
app.Acquisition.Horizontal.HorOffset = 2.0e-7
```

### HorOffsetControl

*Enum*

#### Description

Sets the unit for HorOffset to either Time or Div, using "Time" and "Div", or 0 and 1.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the horizontal offset control to divisions.
app.Acquisition.Horizontal.HorOffsetControl = "Div"
```

#### Values

Div  
Time

**HorOffsetOrigin****Double**

Range: From 0 to 10, step 1

## Description

Sets/Queries the origin, in graticule divisions, of the time scale in which HorOffset is measured. The value 0 corresponds to the left edge of the graticule. The value 10 corresponds to the right edge of the graticule. Requesting a value outside the range will select the nearest allowed value.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the origin of the horizontal trigger offset to 4.0
divisions.
```

**HorScale****DoubleLock step**

Range: From 2e-011 to 10, step 5e-010, locked to 1 2 5

## Description

Sets/Queries the horizontal scale in time per division.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the horizontal acquisition scale to 200 ns/div.
app.Acquisition.Horizontal.HorScale = 2.0e-7
```

**HorUnits****String**

Range: Any number of characters

## Description

Queries the units in which the horizontal scale is measured.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Obtain the units of the horizontal scale.
HorizUnit = app.Acquisition.Horizontal.HorUnits
```

## PART TWO: REFERENCE

---

### MaxSamples

*DoubleLock step*

Range: From 500 to 2.4e+007, step 1000, locked to 1 2.5 5

#### Description

Sets/Queries the maximum permissible number of samples to be used in the acquisition memories. At the faster sample rates, the actual number used can be less than this maximum.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the available memory length per channel to 500
app.Acquisition.Horizontal.MaxSamples = 500
```

### NumPoints

*Integer*

Range: From 2 to 100000000, step 1

#### Description

Queries the number of samples in the current setting of the acquisition memory. For sequence mode, this refers to the number of samples per segment, not to the number in the complete set. Use **MaxSamples** to limit the number of samples acquired.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Obtain the number of points being used in the acquisition
memory.
NumberOfPoints = app.Acquisition.Horizontal.NumPoints
```

### NumSegments

*Integer*

Range: From 2 to 20000, step 1

#### Description

Sets/Queries the number of segments in the sequence mode of acquisition. Only valid when SampleMode = "Sequence".

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Enable sequence mode and capture 500 segments
app.Acquisition.Horizontal.SampleMode = "Sequence"
app.Acquisition.Horizontal.NumSegments = 500
```

**ReferenceClock****Enum**

## Description

Sets/Queries the source of the acquisition reference clock.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the source of the reference clock to External.
app.Acquisition.Horizontal.ReferenceClock = "EXT"
```

## Values

<b>EXT</b>	External reference (use rear-panel BNC)
<b>INT</b>	Internal reference clock

**SampleClock****Enum**

## Description

Sets/Queries the source for the sample clock.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the sample clock to expect an external source.
app.Acquisition.Horizontal.SampleClock = "External"
```

## Values

External  
Internal

**SampleMode****Enum**

## Description

Sets/Queries the mode of acquisition as real-time, sequence, or random interleaved sampling. RIS mode and sequence mode are not available over the entire range of timebases, and are not available simultaneously.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the mode of acquisition to random interleaved sampling.
app.Acquisition.Horizontal.SampleMode = "RIS"
```

## Values

RealTime  
RIS  
Sequence

## PART TWO: REFERENCE

---

### SampleRate

*DoubleLock step*

Range: From 500 to 1e+010, step 1e+008, locked to 1 2.5 5

#### Description

Queries the sample rate of the ADCs. If random interleaved sampling (RIS) is in use, this value will be less than the effective sampling rate of the traces.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the sample rate of the ADCs.
SampleRate = app.Acquisition.Horizontal.SampleRate
```

### SamplingRate

*Double*

Range: From 500 to 1e+010, step (2 digits)

#### Description

Queries the sampling rate. This is the effective sampling rate of the traces, rather than the sampling rate of the ADCs. When random interleaved sampling (RIS) is not in use, both values are the same.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the effective sampling rate of the signal.
SamplingRate = app.Acquisition.Horizontal.SamplingRate
```

### SequenceTimeout

*Double*

Range: From 0.01 to 100, step 0.01

#### Description

Sets/Queries the timeout in segment mode of acquisition if insufficient triggers are received.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the sequence mode timeout to 10 seconds
app.Acquisition.Horizontal.SequenceTimeout = 10.0
```

### SequenceTimeoutEnable

*Bool*

#### Description

Sets/Queries the enabling of the sequence mode timeout.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

## CHAPTER TWO

## Control Reference

```
' Enable the sequence mode timeout.
```

```
app.Acquisition.Horizontal.SequenceTimeoutEnable = True
```

**SmartMemory****Enum**

## Description

Sets the mode of memory management to one of the two modes:

**SetMaximumMemory** – Maximizes the memory length for the given timebase setting; limited by the maximum length that is compatible with the maximum sampling rate that the DSO can achieve.

**FixedSampleRate** – Keeps the sampling rate the same when the timebase is changed; limited by the maximum sampling rate that the DSO can achieve.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the Smart memory mode as fixed sample rate.
app.Acquisition.Horizontal.SmartMemory = "FixedSampleRate"
```

## Values

<b>FixedSampleRate</b>	Maximizes the memory length for the given timebase setting
<b>SetMaximumMemory</b>	Keeps the sampling rate the same when the timebase is changed

**TimePerPoint****Double**

Range: From 1e-012 to 1e+012, step 1e-012

## Description

Queries the time interval between successive samples in the acquisition.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Obtain the time per point of the acquisition.
timePerPt = app.Acquisition.Horizontal.TimePerPoint
MsgBox timePerPt
```

**ZeroDelay****Action**

## Description

Sets the trigger delay to zero, relative to the time origin set by **Horizontal.HorOffsetOrigin**.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the trigger delay to zero, relative to the time origin
' set by Horizontal.HorOffsetOrigin.
app.Acquisition.Horizontal.ZeroDelay
```

## PART TWO: REFERENCE

**TRIGGER***app.Acquisition.Trigger*

This group of CVars controls all aspects of the trigger, except for trigger delay, which is in Acquisition. Horizontal.Names of the form app.Acquisition.Trigger.Sources.xxxx are aliases for simpler names, which are described in this section of the manual. Examples of alias pairs are as follows:

```
app.Acquisition.Trigger.Sources("Cx") = app.Acquisition.Trigger.Cx
app.Acquisition.Trigger.Sources("Ext") = app.Acquisition.Trigger.Ext
app.Acquisition.Trigger.Sources("Line") = app.Acquisition.Trigger.Line
```

**Trigger**

<b>DropoutTime</b>	Double
<b>Glitch</b>	Enum
<b>GlitchHigh</b>	Double
<b>GlitchLow</b>	Double
<b>HoldoffEvents</b>	Integer
<b>HoldoffTime</b>	Double
<b>HoldoffType</b>	Enum
<b>Interval</b>	Enum
<b>IntervalDelta</b>	Double
<b>IntervalHigh</b>	Double
<b>IntervalLow</b>	Double
<b>IntervalNominal</b>	Double
<b>IntervalRange</b>	Enum
<b>PatternType</b>	Enum
<b>QualEvents</b>	Integer
<b>QualFirst</b>	Bool
<b>QualState</b>	Enum
<b>QualTime</b>	Double
<b>QualWait</b>	Enum
<b>Source</b>	Enum
<b>TrigLevel</b>	Double
<b>Type</b>	Enum
<b>ValidateSource</b>	Enum
<b>Width</b>	Enum
<b>WidthDelta</b>	Double
<b>WidthNominal</b>	Double
<b>WidthRange</b>	Enum
<b>ZeroLevel</b>	Action

**DropoutTime****Double**

Range: From 2e-009 to 20, step 5e-010

## Description

Sets/Queries the trigger dropout time. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

app.Acquisition.Trigger.Line is effective only when trigger type is set to "Dropout." See Acquisition.Channels("Cx") for a programming example.

## CHAPTER TWO

## Control Reference

**Glitch****Enum**

## Description

Sets/Queries the type of glitch trigger.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the glitch trigger to trigger on a glitch with duration
between 5 and 10us
app.Acquisition.Trigger.Type = "Glitch"
app.Acquisition.Trigger.Glitch = "InRange"
app.Acquisition.Trigger.GlitchHigh = 10e-6
app.Acquisition.Trigger.GlitchLow = 5e-6
```

## Values

<b>InRange</b>	Trigger on glitch within High/Low range specified
<b>LessThan</b>	Trigger on glitch narrower than limit specified (GlitchHigh)

**GlitchHigh****Double**

Range: From 6e-010 to 20, step 2e-010

## Description

Sets/Queries the upper limit for a glitch trigger. Valid in both the InRange and LessThan glitch trigger types. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

**GlitchLow****Double**

Range: From 6e-010 to 20, step 2e-010

## Description

Sets/Queries the lower limit for a glitch trigger specified as InRange type. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the lower limit for an in-range glitch trigger to 375 ms.
app.Acquisition.Trigger.GlitchLow = 0.375
```

## PART TWO: REFERENCE

---

### HoldoffEvents

*Integer*

Range: From 1 to 1000000000, step 1

#### Description

Sets/Queries the number of events by which the trigger is to be held off. The resolution is 1 at all values.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Enable 'holdoff by events' mode and configure
' to holdoff by 4095 events.
app.Acquisition.Trigger.Type = "Edge"
app.Acquisition.Trigger.HoldoffType = "Events"
app.Acquisition.Trigger.HoldOffEvents = 4095
```

### HoldoffTime

*Double*

Range: From 2e-009 to 20, step 1e-009

#### Description

Sets/Queries the trigger holdoff time. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Enable 'holdoff by time' mode and configure
' to holdoff by 2 seconds.
app.Acquisition.Trigger.Type = "Edge"
app.Acquisition.Trigger.HoldoffType = "Time"
app.Acquisition.Trigger.HoldoffTime = 2.0
```

### HoldoffType

*Enum*

#### Description

Sets/Queries the type of hold-off trigger.

#### Values

<b>Events</b>	Holdoff by events, specified in HoldoffEvents
<b>Off</b>	No Trigger Holdoff
<b>Time</b>	Holdoff by time, specified in HoldoffTime

**Interval****Enum**

## Description

Sets/Queries the interval trigger type. This is only valid when the trigger type is set to Interval.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure the interval trigger, to trigger when the interval
' is within the limits 5us to 10us
app.Acquisition.Trigger.Type = "Interval"
app.Acquisition.Trigger.Interval = "InRange"
app.Acquisition.Trigger.IntervalRange = "Limits"
app.Acquisition.Trigger.IntervalHigh = 10e-6
app.Acquisition.Trigger.IntervalLow = 5e-6
```

## Values

<b>GreaterThan</b>	Trigger when interval is > specified limits
<b>InRange</b>	Trigger when interval is within range
<b>LessThan</b>	Trigger when interval is < specified limits
<b>OutOfRange</b>	Trigger when interval is outside specified range

**IntervalDelta****Double**

Range: From 2e-009 to 20, step 2e-010

## Description

Sets/Queries the tolerance on the pulse interval for an interval trigger specified as InRange type or OutOfRange type. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure the interval trigger, to trigger when the interval
' is 6us +/- 34ns
app.Acquisition.Trigger.Type = "Interval"
app.Acquisition.Trigger.Interval = "InRange"
app.Acquisition.Trigger.IntervalRange = "Delta"
app.Acquisition.Trigger.IntervalDelta = 34e-9
app.Acquisition.Trigger.IntervalNominal = 6e-6
```

## PART TWO: REFERENCE

---

### IntervalHigh

*Double*

Range: From 2e-009 to 20, step 2e-010

Description

Sets/Queries the upper limit for an interval trigger specified as InRange type. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

### IntervalLow

*Double*

Range: From 2e-009 to 20, step 2e-010

Description

Sets/Queries the lower limit for an interval trigger specified as InRange type. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

### IntervalNominal

*Double*

Range: From 4e-009 to 20, step 2e-010

Description

Sets/Queries the nominal pulse interval for an interval trigger specified as InRange type or OutOfRange type. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

Example

See IntervalDelta

### IntervalRange

*Enum*

Description

Sets/Queries the interval trigger setting as either Delta or Limits. Delta is set as a center value with a tolerance. In limits mode the criteria are a lower limit and an upper limit.

Values

<b>Delta</b>	Specify interval as a nominal value and delta.
<b>Limits</b>	Specify interval by a lower and upper limit.

### PatternType

*Enum*

Description

Sets/Queries the pattern (Logic) trigger type.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the pattern trigger type to Nand.
app.Acquisition.Trigger.Type = "Logic"
app.Acquisition.Trigger.PatternType = "Nand"
```

Values

And	Nor
Nand	Or

**CHAPTER TWO****Control Reference****QualEvents****Integer**

Range: From 1 to 99999999, step 1

## Description

Sets/Queries the number of events for which the qualified trigger is to wait. The setting resolution is 1 at all values.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the number of events for trigger holdoff to 8191.
app.Acquisition.Trigger.Type = "Qualify"
app.Acquisition.Trigger.QualWait = "Events"
app.Acquisition.Trigger.QualEvents = 8191
```

**QualFirst****Bool**

## Description

Sets/Queries the state of the "Qualify First" trigger. In sequence mode, this control, when set, allows subsequent triggers to be detected without qualification, after the first segment of a sequence has had a fully qualified trigger. If the control is set to False, every segment has to be qualified individually. When not in sequence mode this control is not used.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the qualified trigger in sequence mode to qualify in
' the first segment only.
app.Acquisition.Horizontal.SampleMode = "Sequence"
app.Acquisition.Trigger.Type = "Qualify"
app.Acquisition.Trigger.QualFirst = True
```

**QualState****Enum**

## Description

Sets/Queries the qualifying state to Above or Below the specified level for the specified input source.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the qualification to Below the specified level.
app.Acquisition.Trigger.QualState = "Below"
```

## Values

Above  
Below

**PART TWO: REFERENCE****QualTime****Double**

Range: From 2e-009 to 20, step 2e-010

## Description

Sets/Queries the threshold time for a qualified trigger. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the tolerance for the interval trigger 144 microseconds.
app.Acquisition.Trigger.QualTime = 1.44e-4
```

**QualWait****Enum**

## Description

Sets/Queries the wait type for qualified trigger.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the qualified trigger to wait for the given time to pass.
app.Acquisition.Trigger.QualWait = "GreaterThan"
```

## Values

Events  
 GreaterThan  
 LessThan  
 Off

**Source****Enum**

## Description

Sets/Queries the trigger source.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the trigger source to external.
app.Acquisition.Trigger.Source = "Ext"
```

## Values

C1	ExtDivide10
C2	ExtTimes
C3	Line
C4	Pattern
Ext	

## CHAPTER TWO

## Control Reference

**TrigLevel****Double**

Range: From -0.25 to 0.25, step 0.0005

## Description

Sets/Queries the trigger level.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the trigger level to 20 mV.
app.Acquisition.Trigger.TrigLevel = 0.020
```

**Type****Enum**

## Description

Sets/Queries the trigger type (mode).

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the trigger type to glitch.
app.Acquisition.Trigger.Type = "Glitch"
```

## Values

- Dropout
- Edge
- Glitch
- Interval
- Logic
- Qualify
- State
- Width

**ValidateSource****Enum**

## Description

Sets/Queries the second trigger source in modes with two sources.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the validation trigger source to C3.
app.Acquisition.Trigger.ValidateSource = "C3"
```

## Values

- C1
- C2
- C3

## PART TWO: REFERENCE

---

C4  
 Ext  
 ExtDivide10  
 ExtTimes10  
 Pattern

### Width

*Enum*

#### Description

Sets/Queries the width trigger type.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure Width Trigger to trigger on a pulse with
' width of 10ns +/- 5ns
app.Acquisition.Trigger.Type = "Width"
app.Acquisition.Trigger.Width = "InRange"
app.Acquisition.Trigger.WidthRange = "Delta"
app.Acquisition.Trigger.WidthNominal = 10e-9
app.Acquisition.Trigger.WidthDelta = 5e-9
```

#### Values

<b>GreaterThan</b>	Trigger when width > specified limit
<b>InRange</b>	Trigger when width is within specified limits
<b>LessThan</b>	Trigger when width < specified limit
<b>OutOfRange</b>	Trigger when width is outside specified range

### WidthDelta

*Double*

Range: From 2e-010 to 20, step 2e-010

#### Description

Sets/Queries the tolerance on pulse width for a width trigger specified as InRange type or OutOfRange type. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the tolerance for the width trigger to 55 ns.
app.Acquisition.Trigger.WidthDelta = 5.5e-8
```

### WidthNominal

*Double*

Range: From 8e-010 to 20, step 2e-010

#### Description

Sets/Queries the nominal pulse width for a width trigger specified as InRange type or OutOfRange type. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the nominal value for the width trigger 55 ns.
app.Acquisition.Trigger.WidthNominal = 5.5e-8
```

**WidthRange****Enum****Description**

Sets/Queries the width setting as either Delta or Limits. When in Delta mode, the controls WidthNominal and WidthDelta are used to specify the nominal value and tolerance. When in Limits mode, the GlitchLow and GlitchHigh controls are used instead.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure Width Trigger to trigger on a pulse with
' width within limits of 5ns and 10ns
app.Acquisition.Trigger.Type = "Width"
app.Acquisition.Trigger.Width = "InRange"
app.Acquisition.Trigger.WidthRange = "Limits"
app.Acquisition.Trigger.GlitchLow = 10e-9
app.Acquisition.Trigger.GlitchHigh = 5e-9
```

**Values**

Delta  
Limits

**ZeroLevel****Action****Description**

Sets the trigger level to zero volts.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the trigger level to zero volts.
app.Acquisition.Trigger.ZeroLevel
```

**PART TWO: REFERENCE****CX*****app.Acquisition.Trigger.Cx***

This group of variables controls triggering from the input channels C1, C2, C3 and C4.

**Cx**

<b>InputImpedance</b>	Enum
<b>Level</b>	Double
<b>PatternState</b>	Enum
<b>Slope</b>	Enum

**InputImpedance*****Enum***

## Description

Reads the input impedance of channel C1, in ohms.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the input impedance of C1 trigger.
ZinC1 = app.Acquisition.Trigger.C1.InputImpedance
MsgBox ZinC1
```

## Values

50

**Level*****Double***

Range: From -0.25 to 0.25, step 0.0005

## Description

Sets/Queries the trigger level for the internal trigger from channel Cx. The setting resolution ranges from about 1.5% to about 2.5%, depending on the numerical value.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the trigger level to 55 mV for triggering on channel C1.
app.Acquisition.Trigger.C1.Level = 0.055
```

**PatternState****Enum**

## Description

Sets/Queries the pattern state for the input channel Cx. Only valid when the trigger mode is set to "Logic."

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the pattern state for channel C1 to low.
app.Acquisition.Trigger.C1.PatternState = "Low"
```

## Values

DontCare  
High  
Low

**Slope****Enum**

## Description

Sets/Queries the direction of the transition to be used for internal triggering from channel Cx.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the direction of the transition as negative
' for triggering on channel C1.
app.Acquisition.Trigger.C1.Slope = "Negative"
```

## Values

Negative  
Positive

**PART TWO: REFERENCE****EXT*****app.Acquisition.Trigger.Ext***

This group of variables controls the external trigger.

**Ext**

<b>Coupling</b>	Enum
<b>InputImpedance</b>	Enum
<b>Level</b>	Double
<b>PatternState</b>	Enum
<b>Slope</b>	Enum

**Coupling*****Enum***

## Description

Sets/Reads the input coupling of the external trigger input.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the input coupling of the external trigger input.
ZinCoupling = app.Acquisition.Trigger.Ext.Coupling
MsgBox ZinCoupling
```

## Values

DC50  
Gnd  
DC 50 ohms

**InputImpedance*****Enum***

## Description

Reads the input impedance of the external trigger.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the input impedance of external trigger input.
ZinExt = app.Acquisition.Trigger.Ext.InputImpedance
MsgBox ZinExt
```

## Values

50

**Level****Double**

Range: From -1 to 1, step 0.001

## Description

Sets/Queries the trigger level for the external trigger.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the trigger level to 55 mV for triggering from
' the external trigger socket.
app.Acquisition.Trigger.Ext.Level = 0.055
```

**PatternState****Enum**

## Description

Sets/Queries the pattern state for the external trigger input.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the pattern state to low for triggering from
' the external trigger socket.
app.Acquisition.Trigger.Ext.PatternState = "Low"
```

## Values

DontCare  
High  
Low

**Slope****Enum**

## Description

Sets/Queries the direction of the transition used for the external trigger.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the direction of the transition as positive for triggering
' from the external trigger socket.
```

## Values

Negative  
Positive

**PART TWO: REFERENCE**

---

**LINE**

*app.Acquisition.Trigger.Line*

This group of variables controls the line trigger: 50 Hz or 60 Hz.

---

**SOURCES***app.Acquisition.Trigger.Sources*

Names of the form `app.Acquisition.Trigger.Sources.xxxx` are aliases for simpler names, which are described in this section of the manual. Examples of alias pairs are as follows:

`app.Acquisition.Trigger.Sources("Cx") = app.Acquisition.Trigger.Cx`

`app.Acquisition.Trigger.Sources("Ext") = app.Acquisition.Trigger.Ext`

`app.Acquisition.Trigger.Sources("Line") = app.Acquisition.Trigger.Line`

## PART TWO: REFERENCE

## CURSORS

*app.Cursors*

This set of variables controls the cursor system.

<b>Pos1</b>	Double
<b>Pos2</b>	Double
<b>Readout</b>	Enum
<b>Track</b>	Bool
<b>Type</b>	Enum
<b>View</b>	Bool
<b>XPos1</b>	Double
<b>XPos2</b>	Double
<b>YPos1</b>	Double
<b>YPos2</b>	Double

**Pos1****Double**

Range: From  $-1.79769e+308$  to  $1.79769e+308$ , step 0

## Description

Sets/Queries the position of the first cursor. For vertical cursors the range is -3.99 to 3.99 divisions. For horizontal cursors the range is from left edge to right edge of the graticule, in the units of the horizontal variable.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the position of the two cursors
app.Cursors.View = "On"
app.Cursors.Type = "HorizRel"
app.Cursors.Pos1 = 50e-9
app.Cursors.Pos2 = -50e-9
```

**Pos2****Double**

Range: From  $-1.79769e+308$  to  $1.79769e+308$ , step 0

## Description

Sets/Queries the position of the second cursor. For vertical cursors the range is -3.99 to 3.99 divisions. For horizontal cursors the range is from left to right of the graticule, in the units of the horizontal variable.

**Readout****Enum**

## Description

Sets/Queries whether the readout of a pair of cursors is Absolute (two separate values), Delta (one value giving the distance between the cursors), or Slope, giving the dv/dt value.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the cursors readout to absolute.
app.Cursors.Readout = "Absolute"
```

## Values

<b>Absolute</b>	Readout each cursor's value
<b>Delta</b>	Readout cursor value differences
<b>Slope</b>	Readout slope, i.e. dv/dt

**Track****Bool**

## Description

Sets/Queries the state of tracking of a pair of cursors. If tracking is enabled, as the first cursor is moved the second will track at a constant distance from it.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set cursors tracking on.
app.Cursors.Track = True
```

**Type****Enum**

## Description

Sets/Queries the currently selected type of cursor.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the cursor type to vertical relative.
app.Cursors.View = "On"
app.Cursors.Type = "VertRel"
```

## Values

<b>HorizAbs</b>	Single cursor, position specified in time
<b>HorizRel</b>	Dual cursors, positions specified in time
<b>VertAbs</b>	Single cursor, position specified in divisions vertically
<b>VertRel</b>	Dual cursors, positions specified in divisions vertically

## PART TWO: REFERENCE

---

### View

**Bool**

#### Description

Sets/Queries visibility of the cursors.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Make the cursors visible.
app.Cursors.View = "On"
```

### XPos1

**Double**

Range: From  $-1.79769e+308$  to  $1.79769e+308$ , step 0

#### Description

Sets/Queries the horizontal position of the first cursor, in the units of the horizontal variable.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the horizontal position of the first cursor to 50 ns.
app.Cursors.XPos1 = 50e-9
```

### XPos2

**Double**

Range: From  $-1.79769e+308$  to  $1.79769e+308$ , step 0

#### Description

Sets/Queries the horizontal position of the second cursor, in the units of the horizontal variable.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the horizontal position of the second cursor to 4.5 ms.
app.Cursors.XPos2 = 4.5e-3
```

### YPos1

**Double**

Range: From  $-3.99$  to  $3.99$ , step 0.01

#### Description

Sets/Queries the vertical position of the first cursor, in graticule divisions.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

**CHAPTER TWO****Control Reference**

---

```
' Set the vertical position of the first cursor.  
app.Cursors.YPos1 = 3.4
```

**YPos2****Double**

Range: From -3.99 to 3.99, step 0.01

**Description**

Sets/Queries the vertical position of the second cursor, in graticule divisions.

**Example**

```
' Visual Basic Script  
Set app = CreateObject("LeCroy.XStreamDSO")  
  
' Set the vertical position of the second cursor.  
app.Cursors.YPos2 = 2.1
```

## PART TWO: REFERENCE

**CUSTOMDSO***app.CustomDSO*

This set of variables controls the CustomDSO system. CustomDSO operates in one of two modes. "Basic" mode, where panel setups can be assigned to named buttons that can be displayed at the bottom of the instrument's display, and "Plug-In" mode, where user-created ActiveX controls can be embedded into the instrument's menu system.

**CustomDSO**

<b>ActionEnable1</b>	Bool
<b>ActionEnable2</b>	Bool
<b>ActionEnable3</b>	Bool
<b>ActionEnable4</b>	Bool
<b>ActionEnable5</b>	Bool
<b>ActionEnable6</b>	Bool
<b>ActionEnable7</b>	Bool
<b>ActionEnable8</b>	Bool
<b>ActionScript1</b>	FileName
<b>ActionScript2</b>	FileName
<b>ActionScript3</b>	FileName
<b>ActionScript4</b>	FileName
<b>ActionScript5</b>	FileName
<b>ActionScript6</b>	FileName
<b>ActionScript7</b>	FileName
<b>ActionScript8</b>	FileName
<b>Mode</b>	Enum
<b>PlugIn1Install</b>	Action
<b>PlugIn1ProgId</b>	String
<b>PlugIn1Remove</b>	Action
<b>PresentAtPowerUp</b>	Bool

**ActionEnable1*****Bool***

## Description

Sets/Queries enabling of the first button in the Basic mode of CustomDSO. Disabled buttons will be grayed out.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Enable the action of button 1 of basic CustomDSO.
app.CustomDSO.ActionEnable1 = True
```

**ActionEnable2*****Bool***

## Description

See ActionEnable1.

## Example

See ActionEnable1.

**CHAPTER TWO****Control Reference****ActionEnable3*****Bool***

Description

See ActionEnable1.

**ActionEnable4*****Bool***

Description

See ActionEnable1.

**ActionEnable5*****Bool***

Description

See ActionEnable1.

**ActionEnable6*****Bool***

Description

See ActionEnable1.

**ActionEnable7*****Bool***

Description

See ActionEnable1.

**ActionEnable8*****Bool***

Description

See ActionEnable1.

**ActionScript1*****FileName***

Range: Any number of characters

Description

Sets/Queries the name of the script file to be recalled by button 1 of basic CustomDSO.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the name of the script file to be recalled by button 1 of
basic CustomDSO.
app.CustomDSO.ActionScript1 =
"C:\LeCroy\XStream\CustomDSO\CustomSet3Script1.lss"
```

**ActionScript2*****FileName***

Range: Any number of characters

Description

See ActionScript1.

## PART TWO: REFERENCE

---

### ActionScript3

**FileName**

Range: Any number of characters

Description

See ActionScript1.

### ActionScript4

**FileName**

Range: Any number of characters

Description

See ActionScript1.

### ActionScript5

**FileName**

Range: Any number of characters

Description

See ActionScript1.

### ActionScript6

**FileName**

Range: Any number of characters

Description

See ActionScript1.

### ActionScript7

**FileName**

Range: Any number of characters

Description

See ActionScript1.

### ActionScript8

**FileName**

Range: Any number of characters

Description

See ActionScript1.

### Mode

**Enum**

Description

Sets/Queries the current mode of Custom DSO.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the CustomDSO mode to Basic
app.CustomDSO.Mode = "Basic"
```

Values

<b>Basic</b>	Basic mode, assign setups to named buttons
<b>Off</b>	CustomDSO Disabled
<b>PlugIn</b>	Plug-In mode, use ActiveX controls to define menu

**CHAPTER TWO****Control Reference****PlugIn1Install****Action**

## Description

Installs the currently nominated COM program for use with the Plug-In mode of Custom DSO.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Install the currently specified plug-in program.
app.CustomDSO.PlugIn1ProgID = "LeCroy.CustomDSODemo.1"
app.CustomDSO.PlugIn1Install
```

**PlugIn1ProgID****String**

Range: Any number of characters

## Description

Sets/Queries the ProgID of the ActiveX control to use during Plug-In mode of CustomDSO. The ProgID "LeCroy.CustomDSODemo.1" can be used to test the system since this simple ActiveX control is installed by default in all X-Stream instruments.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Sets the name of the current plug-in program.
app.CustomDSO.PlugIn1ProgID = "LeCroy.CustomDSODemo.1"
```

**PlugIn1Remove****Action**

## Description

Removes the ActiveX component that is currently in use with the Plug-In mode of Custom DSO.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Remove the currently installed plug-in program.
app.CustomDSO.PlugIn1Remove
```

**PresentAtPowerUp****Bool**

## Description

Sets/Queries whether the Custom DSO menu is to be shown at power up, and when no standard instrument menu is open.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set CustomDSO to be in use at power-up.
app.CustomDSO.PresentAtPowerUp = True
```

## PART TWO: REFERENCE

**DISPLAY***app.Display*

This set of variables controls the properties of the screen display of the instrument.

<b>AxisLabels</b>	Bool
<b>AxisXRotation</b>	Integer
<b>AxisYRotation</b>	Integer
<b>C1Color</b>	Color
<b>C1PrintColor</b>	Color
<b>C2Color</b>	Color
<b>C2PrintColor</b>	Color
<b>C3Color</b>	Color
<b>C3PrintColor</b>	Color
<b>C4Color</b>	Color
<b>C4PrintColor</b>	Color
<b>ClearSweeps</b>	Action
<b>DisplayMode</b>	Enum
<b>F1Color</b>	Color
<b>F1PrintColor</b>	Color
<b>F2Color</b>	Color
<b>F2PrintColor</b>	Color
<b>F3Color</b>	Color
<b>F3PrintColor</b>	Color
<b>F4Color</b>	Color
<b>F4PrintColor</b>	Color
<b>F5Color</b>	Color
<b>F5PrintColor</b>	Color
<b>F6Color</b>	Color
<b>F6PrintColor</b>	Color
<b>F7Color</b>	Color
<b>F7PrintColor</b>	Color
<b>F8Color</b>	Color
<b>F8PrintColor</b>	Color
<b>FactoryDefault</b>	Action
<b>GridIntensity</b>	Integer
<b>GridMode</b>	Enum
<b>GridOnTop</b>	Bool
<b>LockPersistence</b>	Enum
<b>M1Color</b>	Color
<b>M1PrintColor</b>	Color
<b>M2Color</b>	Color
<b>M2PrintColor</b>	Color
<b>M3Color</b>	Color
<b>M3PrintColor</b>	Color
<b>M4Color</b>	Color
<b>M4PrintColor</b>	Color
<b>NumSegmentsDisplayed</b>	Integer
<b>Persist3DQuality</b>	Enum
<b>Persisted</b>	Bool
<b>Persistence3d</b>	Bool

<b>PersistenceLastTrace</b>	Bool
<b>PersistenceMonoChrome</b>	Bool
<b>PersistenceSaturation</b>	Integer
<b>PersistenceStyle</b>	Enum
<b>PersistenceTime</b>	Enum
<b>PreviewPrintColors</b>	Action
<b>ResetAll</b>	Action
<b>SegmentMode</b>	Enum
<b>StartSegment</b>	Integer
<b>TraceStyle</b>	Enum

**AxisLabels****Bool**

## Description

Sets/Queries the visibility of the labels that show the horizontal and vertical limits of each grid.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Show the axis labels.
app.Display.AxisLabels = True
```

**AxisXRotation****Integer**

Range: From -90 to 90, step 1

## Description

Sets/Queries the rotation angle of the 3-D persistence display about the X-axis. The X-axis runs horizontally in the plane of the screen. Positive or negative angles can be used, in the range -90 to +90 degrees. Zero produces a direct plan view if AxisYRotation is also zero.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the rotation about the X axis to 45 degrees.
app.Display.AxisXRotation = 45
```

**AxisYRotation****Integer**

Range: From -90 to 90, step 1

## Description

Sets/Queries the rotation angle of the 3-D persistence display about the Y-axis. The Y-axis runs vertically in the plane of the screen. Positive or negative angles in the range -90 to +90 degrees can be used. A positive angle makes the left side look closer than the right side.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

## PART TWO: REFERENCE

---

```
' Set the rotation about the Y axis to 35 degrees.
```

```
app.Display.AxisYRotation = 35
```

### C1Color

### Color

Range: From 0 to 16777215

#### Description

Sets/Queries the color of trace C1, using a number in the range 0 to FFFFFFFF in hexadecimal. The possible colors are made from any combination of the primary colors, which are set in hexadecimal as Blue = &HFF0000, Green = &HFF00, Red = &HFF. The value may be entered in decimal or in hexadecimal form, though hexadecimal is usually more convenient. If the intensity of a color is to be reduced or increased by a numerical factor, an AND operation must be used afterwards, to prevent corruption of other primary colors.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

Red = &Hff: Green = &H80: Blue = &H00

' Set the color of channel C1 trace to orange
app.Display.C1Color = (Blue * &H10000) + (Green * &H100) + Red
```

### C1PrintColor

### Color

Range: From 0 to 16777215

#### Description

Sets/Queries the color, in the printing palette, of trace C1, using a number in the range 0 to FFFFFFFF in hexadecimal. The primary colors are Blue = &HFF0000, Green = &HFF00, Red = &HFF in hexadecimal. The value may be entered in decimal or in hexadecimal form.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

Red = &Hff: Green = &H80: Blue = &H00

' Set the color of channel C1 trace to orange for printing.
app.Display.C1PrintColor = (Blue * &H10000) + (Green * &H100) +
```

## CHAPTER TWO

## Control Reference

**C2Color****Color**

Range: From 0 to 16777215

Description

See C1Color.

**C2PrintColor****Color**

Range: From 0 to 16777215

Description

See C1Printcolor.

**C3Color****Color**

Range: From 0 to 16777215

Description

See C1Color.

**C3PrintColor****Color**

Range: From 0 to 16777215

Description

See C1Printcolor.

**C4Color****Color**

Range: From 0 to 16777215

Description

See C1Color.

**C4PrintColor****Color**

Range: From 0 to 16777215

Description

See C1Printcolor.

**ClearSweeps****Action**

Description

Initiates a Clear Sweeps operation. Clears history only for persistence traces, see the main Clear Sweeps control **app.ClearSweeps**, or the ClearSweeps control in other subsystems for other options.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Initiate a clear sweeps action for persistence traces.
app.Display.ClearSweeps
```

## PART TWO: REFERENCE

---

### DisplayMode

*Enum*

#### Description

Sets/Queries the display mode as either "Scope," showing the normal instrument screen, or "WebEdit," showing the web processor editing panel. WebEdit mode is available only with certain software options, including XMATH and XMAP.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Switch to WebEdit mode
app.Display.DisplayMode = "WebEdit"
```

#### Values

Scope  
WebEdit

### F1Color

*Color*

Range: From 0 to 16777215

#### Description

See C1Color.

### F1PrintColor

*Color*

Range: From 0 to 16777215

#### Description

See C1Printcolor.

### F2Color

*Color*

Range: From 0 to 16777215

#### Description

See C1Color.

### F2PrintColor

*Color*

Range: From 0 to 16777215

#### Description

See C1Printcolor.

### F3Color

*Color*

Range: From 0 to 16777215

#### Description

See C1Color.

### F3PrintColor

*Color*

Range: From 0 to 16777215

#### Description

See C1Printcolor.

**CHAPTER TWO****Control Reference**

---

**F4Color****Color**

Range: From 0 to 16777215

Description

See C1Color.

**F4PrintColor****Color**

Range: From 0 to 16777215

Description

See C1Printcolor.

**F5Color****Color**

Range: From 0 to 16777215

Description

See C1Color.

**F5PrintColor****Color**

Range: From 0 to 16777215

Description

See C1Printcolor.

**F6Color****Color**

Range: From 0 to 16777215

Description

See C1Color.

**F6PrintColor****Color**

Range: From 0 to 16777215

Description

See C1Printcolor.

**F7Color****Color**

Range: From 0 to 16777215

Description

See C1Color.

**F7PrintColor****Color**

Range: From 0 to 16777215

Description

See C1Printcolor.

**F8Color****Color**

Range: From 0 to 16777215

Description

See C1Color.

## PART TWO: REFERENCE

---

### F8PrintColor

*Color*

Range: From 0 to 16777215

Description

See C1Printcolor.

### FactoryDefault

*Action*

Description

Restores the display of the instrument to the factory default settings.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Restore the display to the factory pre-set state.
app.Display.FactoryDefault
```

### GridIntensity

*Integer*

Range: From 0 to 100, step 1

Description

Sets/Queries the grid intensity as a percentage of the maximum value, with a resolution of 1%.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the grid intensity to 60% of the maximum.
app.Display.GridIntensity = 60
```

### GridMode

*Enum*

Description

Sets/Queries the grid mode. For example, the commands "Single" and "Dual" set the grid mode until countermanded. "Auto" allows the instrument to set the grid mode most suitable for the current number of visible traces.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Enter Octal grid mode
app.Display.GridMode = "Octal"
```

Values

<b>Auto</b>	Automatically choose grid mode, one trace per grid
<b>Dual</b>	Dual grid mode
<b>Octal</b>	Octal grid mode

<b>Quad</b>	Quad grid mode
<b>Single</b>	Single grid mode
<b>XY</b>	XY grid mode
<b>XYDual</b>	XY + Dual grid mode
<b>XYSingle</b>	XY + Single grid mode

**GridOnTop****Bool**

## Description

Sets/Queries whether the grid lines lie over the traces or vice versa.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the grid lines to be over the trace lines.
app.Display.GridOnTop = True
```

**LockPersistence****Enum**

## Description

Sets/Queries whether the persistence states of the visible traces are locked together or separate.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the persistence display to per trace, not locked.
app.Display.LockPersistence = "PerTrace"
```

## Values

AllLocked  
PerTrace

**M1Color****Color**

Range: From 0 to 16777215

## Description

See C1Color.

**M1PrintColor****Color**

Range: From 0 to 16777215

## Description

See C1Printcolor.

**M2Color****Color**

Range: From 0 to 16777215

## Description

See C1Color.

**PART TWO: REFERENCE**

---

**M2PrintColor****Color**

Range: From 0 to 16777215

Description

See C1Printcolor.

**M3Color****Color**

Range: From 0 to 16777215

Description

See C1Color.

**M3PrintColor****Color**

Range: From 0 to 16777215

Description

See C1Printcolor.

**M4Color****Color**

Range: From 0 to 16777215

Description

See C1Color.

**M4PrintColor****Color**

Range: From 0 to 16777215

Description

See C1Printcolor.

**NumSegmentsDisplayed****Integer**

Range: From 1 to 80, step 1

Description

In sequence mode, sets/queries the number of segments displayed on the screen.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the number of displayed segments to 20
app.Acquisition.Horizontal.SampleMode = "Sequence"
app.Acquisition.Horizontal.NumSegments = 20
app.Display.NumSegmentsDisplayed = 20
app.Display.SegmentMode = "Mosaic"
```

**Persist3DQuality****Enum**

Description

Sets/Queries the type of 3-D plot that is displayed.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the type of the 3-D persistence plot.
app.Display.Persist3DQuality = "WireFrame"
```

## Values

```
Shaded
Solid
WireFrame
```

**Persisted****Bool**

## Description

Sets/Queries whether persistence mode is in use. If the previously set persistence mode is per trace, the persisted cvar will be set to True by this command, even if none of the traces has been set to persistence.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the state of persistence mode.
Persist = app.Display.Persisted
```

**Persistence3d****Bool**

## Description

Sets/Queries whether the persistence 3-D mode is activated.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the 3-D display to off.
app.Display.Persistence3d = False
```

**PersistenceLastTrace****Bool**

## Description

Sets/Queries whether the last created trace is shown over the persistence trace.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the persistence display to show the last trace
' on top of the persistence trace.
app.Display.PersistenceLastTrace = True
```

**PART TWO: REFERENCE****PersistenceMonoChrome*****Bool***

## Description

Sets/Queries whether the persistence mode is monochrome.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the persistence mode as color.
app.Display.PersistenceMonoChrome = False
```

**PersistenceSaturation*****Integer***

Range: From 0 to 100, step 1

## Description

Sets/Queries the population level, relative to the maximum possible level, at which the persistence traces reach maximum intensity, and above which there are no further changes in color or intensity.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the persistence saturation level to 60%.
app.Display.PersistenceSaturation = 60
```

**PersistenceStyle*****Enum***

## Description

Sets/Queries the type of persistence trace displayed.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the persistence style to color graded.
app.Display.PersistenceStyle = "ColorGraded"
```

## Values

3d  
Analog  
ColorGraded

**PersistenceTime*****Enum***

## Description

Sets/Queries decay time for trace persistence, expressed as a number of seconds or as infinite.

## Example

```
' Visual Basic Script
```

## CHAPTER TWO

## Control Reference

```
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Set the persistence time to 10 seconds.
```

```
app.Display.PersistenceTime = "10s"
```

**Values**

```
0.5s
```

```
10s
```

```
1s
```

```
20s
```

```
2s
```

```
5s
```

```
Infinite
```

**PreviewPrintColors****Action****Description**

Shows the instrument display in the current color scheme selected for printing.

**Example**

```
' Visual Basic Script
```

```
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Show the current color scheme selected for printing.
```

```
app.Display.PreviewPrintColors
```

**ResetAll****Action****Description**

Turns off persistence on all traces where it has been turned on.

**Example**

```
' Visual Basic Script
```

```
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Reset all persistence traces to non-persisted mode.
```

```
app.Display.ResetAll
```

**SegmentMode****Enum****Description**

Sets/Queries the display mode for segmented input channels. All visible channels are set to the same display mode by a single command.

**Example**

```
' Visual Basic Script
```

```
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Set the display mode for segments in C2 to perspective.
```

```
app.Acquisition.C2.SegmentMode = "Perspective"
```

## PART TWO: REFERENCE

---

Values

<b>Adjacent</b>	All segments displayed end-to-end, left to right
<b>Mosaic</b>	Segments displayed in a mosaic, top-left to bottom right
<b>Overlay</b>	Segments are overlaid, similar to persistence
<b>Perspective</b>	Segments are displayed in a perspective view
<b>Waterfall</b>	Successive segments are displayed with increasing vertical offset

### StartSegment

*Integer*

Range: From 1 to 100000, step 1

Description

Sets/Queries the selection of the first segment to be shown on the screen.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the first shown segment to 10.
app.Display.StartSegment = 10
```

### TraceStyle

*Enum*

Description

Sets/Queries the style in which traces are drawn.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the state of the persistence mode.
TraceStyle = app.Display.TraceStyle
```

Values

<b>Line</b>	Connect adjacent samples with straight lines
<b>Points</b>	Show only the sample points

**HARDCOPY***app.HardCopy*

This set of variables controls the transfer of information about the screen display to destinations such as disk files, internal memories, printers, and remote computers.

<b>Destination</b>	Enum
<b>Directory</b>	FileName
<b>E-MailMessage</b>	String
<b>GridAreaOnly</b>	Bool
<b>HardcopyArea</b>	Enum
<b>ImageFileFormat</b>	Enum
<b>Orientation</b>	Enum
<b>PreferredFilename</b>	String
<b>Print</b>	Action
<b>PromptForMessage</b>	Bool
<b>SelectedPrinter</b>	Enum
<b>StripChart</b>	Bool
<b>StripChartFactor</b>	Enum
<b>UsePrintPalette</b>	Bool

**Destination***Enum*

Description

Sets/Queries the destination for hard copy.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the destination for hard copy to e-mail.
app.Hardcopy.Destination = "EMail"
```

Values

<b>Clipboard</b>	Send to clipboard for pasting into other applications
<b>E-Mail</b>	Send image in an E-Mail
<b>File</b>	Store image in a file
<b>Printer</b>	Print to a local, or networked printer

**Directory***FileName*

Range: Any number of characters

Description

Sets/Queries the directory for hard copy to a file.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the directory for hard copy to files as "D:\HC"
app.Hardcopy.Directory = "D:\HC"
```

**PART TWO: REFERENCE****EMailMessage****String**

Range: Any number of characters

## Description

Sets/Queries the e-mail message.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Create the e-mail message - "Results for run 89".
app.Hardcopy.EMailMessage = "Results for run 89"
```

**GridAreaOnly****Bool**

## Description

Sets/Queries whether hard copy is of grid area only.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the status of Grid Area Only.
GridArea = app.Hardcopy.GridAreaOnly
```

**HardcopyArea****Enum**

## Description

Sets/Queries the area of the screen to be included in a hard copy.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Select the DSO screen area for hard copy.
app.Hardcopy.HardCopyArea = "DSOWindow"
```

## Values

<b>DSOWindow</b>	Include only the DSO window
<b>FullScreen</b>	Include the full display screen
<b>GridAreaOnly</b>	Include the grid area only (doesn't include menus)

## CHAPTER TWO

## Control Reference

**ImageFileFormat****Enum**

## Description

Sets/Queries the file format for hard copy data.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Select the format PNG for a file of hard copy data.
app.Hardcopy.Destination = "File"
app.Hardcopy.ImageFileFormat = "PNG"
```

## Values

<b>BMP</b>	Windows Bitmap
<b>BMPCOMP</b>	8-bit Windows Bitmap
<b>JPEG</b>	JPEG - JFIF Compliant
<b>PNG</b>	Portable Network Graphics
<b>PSD</b>	Adobe Photoshop 3.0
<b>TIFF</b>	Tagged Image File Format

**Orientation****Enum**

## Description

Sets/Queries the orientation for hard copy to landscape. Valid only when outputting to a printer as opposed to a file, the clipboard, or an e-mail.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the orientation for hardcopy to landscape.
app.Hardcopy.Orientation = "Landscape"
```

## Values

Landscape  
Portrait

**PreferredFilename****String**

Range: Any number of characters

## Description

Sets/Queries the preferred filename to use for hard copy.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the preferred filename to PrintFile.
app.Hardcopy.PreferredFilename = "PrintFile"
```

**PART TWO: REFERENCE****Print****Action**

## Description

Initiates a hard copy.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Initiate a hard copy.
app.Hardcopy.Print
```

**PromptForMessage****Bool**

## Description

Sets/Queries whether a prompt will be given when screen dump to e-mail is requested. The prompt offers the possibility of sending a message with the e-mail.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Enable the prompt for message with e-mail.
app.Hardcopy.PromptForMessage = True
```

**SelectedPrinter****Enum**

## Description

Sets/Queries the selection of the printer for hard copy. White space and punctuation are removed from the string.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Select BarbondaleTintJet as the printer for hardcopy
app.Hardcopy.SelectedPrinter = "BarbondaleTintJet"
```

**StripChart****Bool**

## Description

Sets/Queries the status of strip chart mode of printing. Valid only when outputting to the internal printer.

**StripChartFactor****Enum**

## Description

Sets/Queries the scale factor for strip chart printing. Valid only when outputting to the internal printer.

## Example

```
' Visual Basic Script
```

```
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Set the strip chart scale to 5 cm/division.  
app.Hardcopy.StripChartFactor = "5cmdiv"
```

**Values**

```
100cmdiv  
10cmdiv  
1cmdiv  
200cmdiv  
20cmdiv  
2cmdiv  
50cmdiv  
5cmdiv
```

**UsePrintPalette****Bool****Description**

Sets/Queries the status of the color palette for hard copy. The screen palette or the print palette can be used. The print palette defaults to a white background, but keeps the approximate colors of all other objects. This mode saves ink/toner when printing onto paper.

**Example**

```
' Visual Basic Script  
Set app = CreateObject("LeCroy.XStreamDSO")  
  
' Ignore the special print color palette for hard copy.  
app.Hardcopy.UsePrintPalette = False
```

## PART TWO: REFERENCE

## MATH

*app.Math*

Variables of the form `app.Math.xxxx` control the mathematical functions F1 through F8.

Names of the form `app.Math.Functions("Fx").xxxx` are aliases for simpler names, which are described in this section of the manual. Examples of alias pairs are as follows:

```
app.Math.Functions("Fx") = app.Math.Fx
```

```
app.Math.Functions("Fx").Out.Result = app.Math.Fx.Out.Result
```

```
app.Math.Functions("Fx").Zoom = app.Math.Zoom.Fx
```

<b>ClearSweeps</b>	Action
<b>ResetZoom</b>	Action

**ClearSweeps****Action**

## Description

Clears sweeps for history functions such as average, histogram, and trend. See also the general **app.ClearSweeps** control which clears accumulated data for all subsystems, including persistence.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Clear sweeps for all history functions.
app.Math.ClearSweeps
```

**ResetZoom****Action**

## Description

Resets zoom to its default settings.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Reset zoom.
app.Math.ResetZoom
```

**FUNCTIONS***app.Math.Functions*

Names of the form `app.Math.Functions("Fx").xxxx` are aliases for simpler names, which are described in the section of the manual devoted to `app.Math`. Examples of alias pairs are as follows:

```
app.Math.Functions("Fx") = app.Math.Fx
```

```
app.Math.Functions("Fx").Out.Result = app.Math.Fx.Out.Result
```

```
app.Math.Functions("Fx").Zoom = app.Math.Zoom.Fx
```

See `Acquisition.Channels` for a programming example.

**FX***app.Math.Fx*

This set of variables controls the math functions F1 through F8.

<b>AxisXRotation</b>	Integer
<b>AxisYRotation</b>	Integer
<b>ClearSweeps</b>	Action
<b>DoResetZoom</b>	Action
<b>DoStoreToMemoryTrace</b>	Action
<b>Equation</b>	String
<b>GraphOp</b>	Enum
<b>LabelsPosition</b>	String
<b>LabelsText</b>	String
<b>MathMode</b>	Enum
<b>MeasureOp</b>	Enum
<b>Operator1</b>	Enum
<b>Operator2</b>	Enum
<b>Persist3DQuality</b>	Enum
<b>Persisted</b>	Bool
<b>Persistence3d</b>	Bool
<b>PersistenceMonoChrome</b>	Bool
<b>PersistenceSaturation</b>	Integer
<b>PersistenceTime</b>	Enum
<b>ShowLastTrace</b>	Bool
<b>Source1</b>	Enum
<b>Source2</b>	Enum
<b>Source3</b>	Enum
<b>UseDotJoin</b>	Bool
<b>UseGrid</b>	String
<b>View</b>	Bool
<b>ViewLabels</b>	Bool

**AxisXRotation***Integer*

Range: From -90 to 90, step 1

## Description

Sets/Queries the state of the X-axis rotation control, used only in 3-D persistence mode to control the apparent viewing position.

## Example

```
' Visual Basic Script
```

**PART TWO: REFERENCE**

---

```
Set app = CreateObject("LeCroy.XStreamDSO")
```

Set the rotation about the X-axis to 35 degrees for trace F3.

```
app.Acquisition.F3.AxisXRotation = 35
```

**AxisYRotation****Integer**

Range: From -90 to 90, step 1

## Description

Sets/Queries the state of the Y-axis rotation control, used only in 3-D persistence mode to control the apparent viewing position.

## Example

```
' Visual Basic Script
```

```
Set app = CreateObject("LeCroy.XStreamDSO")
```

Set the rotation about the Y-axis to 25 degrees for trace F3.

```
app.Acquisition.F3.AxisYRotation = 25
```

**ClearSweeps****Action**

## Description

Clears accumulated data for a single function trace.

## Example

```
' Visual Basic Script
```

```
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Reset accumulation for trace F1
```

```
app.Math.F1.ClearSweeps
```

**DoResetZoom****Action**

## Description

Resets the zoom state of math trace Fx.

## Example

```
' Visual Basic Script
```

```
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Reset zoom of math function F3.
```

```
app.Math.F3.DoResetZoom
```

**DoStoreToMemoryTrace****Action**

## Description

Stores data from math function Fx to a memory trace.

Destination for F1 will be M1, F2 will be M2, etc.

## CHAPTER TWO

## Control Reference

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Store math function F2 to a memory trace.
app.Math.F2.DoStoreToMemoryTrace
```

## Equation

*String*

Range: Any number of characters.

## Description

Queries the equation that defines the math function Fx.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the definition of math function F3.
EquationF3 = app.Math.F3.Equation
MsgBox EquationF3
```

## GraphOp

*Enum*

## Description

Sets/Queries the graph operator of math function Fx. This can be used to produce a Histogram, Track, or Trend of a selected measurement directly within the Math subsystem, without the need to use the measurement subsystem. This control is only valid when MathMode is set to "Graphing."

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to show the histogram of the amplitude of C1
app.Math.F1.View = True
app.Math.F1.MathMode = "Graphing"
app.Math.F1.GraphOp = "Histogram"
app.Math.F1.MeasureOp = "Amplitude"
app.Math.F1.Source1 = "C1"
```

## Values

Histogram  
Track  
Trend

## LabelsPosition

*String*

Range: Any number of characters.

## PART TWO: REFERENCE

---

### Description

Sets/Queries the horizontal position of the label attached to the acquisition trace Fx. The unit of measure is the unit of the horizontal scale. Measurement is made from the trigger point.

This control is a string, not a numeric value, which allows multiple labels to be positioned, as shown in the following example.

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Add a couple of labels to trace F1, one at 0ns, and one at 55ns

app.SetToDefaultSetup
app.Math.F1.View = True
app.Math.F1.ViewLabels = True
app.Math.F1.LabelsPosition = "0.0, 55e-9"
```

### LabelsText

**String**

Range: Any number of characters.

### Description

Sets/Queries the text that appears in labels attached to acquisition trace Cx. Multiple labels can be specified by using a comma as a delimiter.

### MathMode

**Enum**

### Description

Sets/Queries the math mode.

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the mode of the math function F1
app.Math.F1.MathMode = "TwoOperators"
```

### Values

<b>Graphing</b>	Graphing mode, chain a measurement and a graphing operator
<b>OneOperator</b>	Single math operator
<b>TwoOperators</b>	Chain two math operators
<b>WebEdit</b>	Measurement is defined using the Processing Web Editor

### MeasureOp

**Enum**

### Description

Sets/Queries the measure operator of math function Fx. Valid only when MathMode = "Graph".

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the measure operator of math function F1
app.Math.F1.View = True
app.Math.F1.MathMode = "Graph"
app.Math.F1.GraphOp = "Histogram"
app.Math.F1.MeasureOp = "Amplitude"
```

## Values

Amplitude  
Area  
Base  
Cycles  
Delay  
DeltaDelay  
DeltaPeriodAtLevel  
DeltaTimeAtLevel  
DeltaWidthAtLevel  
Duration  
DutyAtLevel  
DutyCycle  
EdgeAtLevel  
ExcelParam  
ExtinctionRatio  
EyeAmplitude  
EyeAvgPower  
EyeBER  
eyecrossing  
EyeHeight  
EyeOneLevel  
EyeQ  
EyeWidth  
EyeZeroLevel  
Fall  
Fall8020  
FallAtLevel  
FirstPoint  
Frequency  
FrequencyAtLevel  
FullWidthAtHalfMaximum  
FullWidthAtXX  
HalfPeriod  
HistogramAmplitude  
HistogramBase  
HistogramMaximum  
HistogramMean  
HistogramMedian  
HistogramMinimum

## PART TWO: REFERENCE

---

HistogramRms  
 HistogramSdev  
 HistogramTop  
 HoldTime  
 LastPoint  
 LevelAtX  
 MathcadParam  
 MATLABParameter  
 Maximum  
 MaximumPopulation  
 Mean  
 Median  
 Minimum  
 Mode  
 NarrowBandPhase  
 NarrowBandPower  
 npoints  
 Null  
 NumberOfModes  
 OvershootNegative  
 OvershootPositive  
 ParamScript  
 Peaks  
 PeakToPeak  
 Percentile  
 Period  
 PeriodAtLevel  
 Phase  
 PopulationAtX  
 Range:  
 Rise  
 Rise2080  
 RiseAtLevel  
 RootMeanSquare  
 Setup  
 Skew  
 StandardDeviation  
 TIE  
 TimeAtLevel  
 Top  
 TotalPopulation  
 Width  
 WidthAtLevel  
 XAtMaximum  
 XAtMinimum  
 XAtPeak

### Operator1

### Enum

#### Description

Sets/Queries the first operator of math function Fx. When MathMode = "OneOperator", this is the only math operator, when MathMode = "TwoOperators", this is the first of two operators. When MathMode = "Graph", this control has no effect. Note also that the list of available math operators varies, depending upon the instrument

**CHAPTER TWO****Control Reference**

---

model number and the list of installed software.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Define the first operator of math function F1 as an Average
app.Math.F1.View = True
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Average"
```

**Values**

AbsoluteValue  
Average  
Boxcar  
Copy  
Correlation  
Derivative  
Deskew  
Difference  
EnhancedResolution  
Envelope  
ExcelMath  
Exp  
Exp10  
FFT  
Filter  
Floor  
Histogram  
Integral  
Interpolate  
Invert  
Ln  
Log10  
MathcadMath  
MATLABWaveform  
Null  
PersistenceHistogram  
PersistenceTraceMean  
PersistenceTraceRange  
PersistenceTraceSigma  
Product  
Ratio  
Reciprocal  
Rescale  
options.Roof  
SegmentSelect  
SinXOverX  
Sparse  
Square  
SquareRoot

## PART TWO: REFERENCE

---

Sum  
Track  
Trend  
WaveScript  
Zoom

### Operator2

*Enum*

#### Description

Sets/Queries the second operator of math function Fx. This control is valid only when MathMode = "TwoOperators", indicating that two math operators are chained to produce a single result.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Define the first operator of math function F1 as an Average
app.Math.F1.View = True
app.Math.F1.MathMode = "TwoOperators"
app.Math.F1.Operator1 = "Invert"
app.Math.F1.Operator2 = "Average"
```

#### Values

AbsoluteValue  
Average  
Boxcar  
Copy  
Correlation  
Derivative  
Deskew  
Difference  
EnhancedResolution  
Envelope  
ExcelMath  
Exp  
Exp10  
FFT  
Filter  
Floor  
Histogram  
Integral  
Interpolate  
Invert  
Ln  
Log10  
MathcadMath  
MATLABWaveform  
Null  
PersistenceHistogram  
PersistenceTraceMean  
PersistenceTraceRange

PersistenceTraceSigma  
 Product  
 Ratio  
 Reciprocal  
 Rescale  
 Roof  
 SegmentSelect  
 SinXOverX  
 Sparse  
 Square  
 SquareRoot  
 Sum  
 Track  
 Trend  
 WaveScript  
 Zoom

**Persist3DQuality****Enum**

## Description

Sets/Queries the state of the 3-D Persistence quality control. Controls the way that the persistence trace is rendered.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set persistence 3-D to shaded for trace F1
app.Math.F1.Persist3DQuality = "Shaded"
```

## Values

Shaded  
 Solid  
 WireFrame

**Persisted****Bool**

## Description

Sets/Queries the persisted state of the function waveform. If the Display.LockPersistence control is set to "AllLocked," the persisted state of all displayed waveforms will be the same. If the Display.LockPersistence control is set to "PerTrace," the persisted state of each waveform can be independently controlled.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set persistence on for trace F3.
app.Math.F3.Persisted = True
```

**PART TWO: REFERENCE****Persistence3d****Bool**

## Description

Sets/Queries the 3-D persistence state. When True, the persistence display for this channel will be displayed as a three dimensional surface map.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

Set persistence plot as 3-D for trace F4.
app.Acquisition.F4.Persistence3D = True
```

**PersistenceMonoChrome****Bool**

## Description

Sets/Queries the monochrome persistence state. When True, the persistence display for this channel will be monochromatic, whether 2-D or 3-D.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set persistence monochrome on for trace F1
app.Math.F1.PersistenceMonoChrome = True
```

**PersistenceSaturation****Integer**

Range: From 1 to 100, step 1

## Description

Sets/Queries the saturation threshold for persisted waveforms. All information at this level or higher will be recorded with the same color or intensity.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the persistence saturation level for trace F1.
app.Math.F1.PersistenceSaturation = 40
```

**PersistenceTime****Enum**

## Description

Sets/Queries the state of the Persistence Time control. Controls the persistence decay time for this trace.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Set the persistence time for the trace F1 to 10 seconds.
app.Math.F1.PersistenceTime = "10s"
```

**Values**

```
0.5s
10s
1s
20s
2s
5s
Infinite
```

**ShowLastTrace****Bool****Description**

Sets/Queries the state of the Show Last Trace control. If True, then when this trace is displayed in persistence mode, the last acquired waveform will be superimposed on the accumulating persistence map. See the general description above for a discussion of the locked and unlocked persistence modes.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Do not show the last trace for the persistence trace of trace
F1.
```

**Source1****Enum****Description**

Sets/Queries the first source of the first operator in Fx. The two possible sources of Operator1 are Source1 and Source2. Source3 is the second source to Operator2, with the first source of Operator2 being the output of Operator1. The list of available sources depends on the instrument model and its installed software options.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Define the first source of math function F1 as C3.
app.Math.F1.Source1 = "C3"
```

**Values**

```
BadBits
Bits
C1
C2
C3
C4
Eye
```

**PART TWO: REFERENCE**

---

F2  
 F3  
 F4  
 F5  
 F6  
 F7  
 F8  
 M1  
 M2  
 M3  
 M4  
 P1  
 P2  
 P3  
 P4  
 P5  
 P6  
 P7  
 P8  
 PRBS

**Source2****Enum**

## Description

Sets/Queries the second source of the first operator in Fx. The list of available sources depends on the instrument model and its installed software options.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Define the second source of math function F2 as C4.
app.Math.F1.Source2 = "C4"
```

## Values

BadBits  
 Bits  
 C1  
 C2  
 C3  
 C4  
 Eye  
 F2  
 F3  
 F4  
 F5  
 F6  
 F7  
 F8  
 M1  
 M2  
 M3  
 M4

None  
PRBS

**Source3***Enum*

## Description

Sets/Queries the second source of the second operator in Fx. Only valid when MathMode = "Dual". The list of available sources depends on the instrument model and its installed software options.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Define the third source of math function F1 as C1.
app.Math.F1.Source3 = "C1"
```

## Values

BadBits  
Bits  
C1  
C2  
C3  
C4  
Eye  
F2  
F3  
F4  
F5  
F6  
F7  
F8  
M1  
M2  
M3  
M4  
None  
PRBS

**UseDotJoin***Bool*

## Description

Sets/Queries the DotJoin state of the trace. When True, straight line segments will be drawn between sample points. When False, only the sample points will be shown. See **Display.TraceStyle** for a control that can change the setting for all displayed traces simultaneously.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Turn dot join off for trace F1.
app.Math.F1.UseDotJoin = False
```

## PART TWO: REFERENCE

---

### UseGrid

**String**

Range: Any number of characters.

Description

Sets/Queries the grid in use for the math trace Fx. See also **app.Acquisition.Cx.UseGrid**.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Place math trace F3 on grid YT4.
app.Math.F3.UseGrid = "YT4"
```

### View

**Bool**

Description

Sets/Queries whether the trace of math function Fx is visible. Even when math traces are not visible, but are being used as inputs to other math functions and/or measurements, they are computed.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Show math trace F3.
app.Math.F3.View = True
```

### ViewLabels

**Bool**

Description

Sets/Queries whether trace labels, defined with LabelsText and LabelsPosition controls, are shown.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Show the user-defined trace label for trace F1
app.Math.F1.ViewLabels = True
```

---

**OPERATOR1SETUP***app.Math.Fx.Operator1Setup*

This node is dynamically created, and will contain the controls for the operator currently selected into Operator1. See the Math/Measure Control reference at the end of this manual for a list of these controls.

## PART TWO: REFERENCE

---

### RESULT

### app.Math.Fx.Out.Result

Properties of the type `xxxx.Out.Result.yyyy` are those of the last completed acquisition. They are not affected if other CVars are changed after that acquisition was completed. This distinction between "Out.Result" properties and other CVars is most important when the trigger mode is Single or Stopped. You should treat "Out.Result" properties as read-only.

See **app.Acquisition.C1.Out.Result** for a detailed description of all properties available for the output of a Math function.

**ZOOM***app.Math.Fx.Zoom*

This set of variables controls the zoom functions for math trace Fx.

**Zoom**

<b>HorPos</b>	Double
<b>HorZoom</b>	Double
<b>ResetZoom</b>	Action
<b>VariableHorZoom</b>	Bool
<b>VariableVerZoom</b>	Bool
<b>VerPos</b>	Double
<b>VerZoom</b>	Double

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Turn on trace F1, will default to Zoom-Only
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"

' Zoom trace F1 by a factor of 2 horizontally and vertically
app.Math.F1.Zoom.ResetZoom
app.Math.F1.Zoom.HorZoom = 2.0
app.Math.F1.Zoom.VerZoom = 2.0
```

**HorPos****Double**

Range: From -0.5 to 0.5, step (8 digits)

**Description**

Sets/Queries the horizontal position of the center of the grid on the zoomed trace Fx. The unit of measure is the screen width, that is, 0.3 means a shift of three of the ten divisions. A positive value moves the trace to the left.

**HorZoom****Double**

Range: From 0.1 to 1e+006, step (8 digits)

**Description**

Sets/Queries the horizontal magnification of the trace Fx. The magnification will be in a 1 2 5 10 sequence unless variable horizontal magnification has been set.

**ResetZoom****Action****Description**

Resets the trace Fx to x1 zoom and zero offset in both axes, so that it is identical to its input trace.

**PART TWO: REFERENCE**

---

**VariableHorZoom****Bool**

## Description

Sets/Queries the ability to zoom horizontally by a continuously variable factor. If a horizontal zoom of 0.9 is set, while variable zoom is off, the horizontal zoom will be set to 1.0. If the variable zoom is then enabled, the factor of 0.9 will have been remembered, and it will be used.

**VariableVerZoom****Bool**

## Description

Sets/Queries the ability to zoom vertically by a continuously variable factor. If a vertical zoom of 0.9 is set, while variable zoom is off, the vertical zoom will be set to 1.0. If the variable zoom is then enabled, the factor of 0.9 will have been remembered, and it will be used.

**VerPos****Double**

Range: From -1.5 to 1.5, step (8 digits)

## Description

Sets/Queries the vertical position of the center of the grid on the zoomed trace Fx. The unit of measure is the screen height, that is, 0.375 means a shift of three of the eight divisions. A positive value moves the trace downwards.

**VerZoom****Double**

Range: From 0.1 to 100, step (8 digits)

## Description

Sets/Queries the vertical magnification of the trace Fx. The magnification will be in a 1 2 5 10 sequence unless VariableVerZoom has been set to True, in which case it will be continuously variable.

This set of variables controls the display of data in X-vs.-Y mode. Only valid when the instrument is in XY, XYSingle, or XYDual display modes.

<b>AxisXRotation</b>	Integer
<b>AxisYRotation</b>	Integer
<b>ClearSweeps</b>	Action
<b>InputX</b>	Enum
<b>InputY</b>	Enum
<b>Persist3DQuality</b>	Enum
<b>Persisted</b>	Bool
<b>Persistence3d</b>	Bool
<b>PersistenceMonoChrome</b>	Bool
<b>PersistenceSaturation</b>	Integer
<b>PersistenceTime</b>	Enum
<b>ShowLastTrace</b>	Bool
<b>UseDotJoin</b>	Bool

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Switch to XY+Dual Grid Mode
app.Display.GridMode = "XYDual"

' Configure XY to show C1 vs. C2 in 3D Persistence mode
app.Math.XY.InputX = "C1"
app.Math.XY.InputY = "C2"
app.Math.XY.Persisted = True
app.Math.XY.Persistence3d = True
app.Math.XY.PersistenceMonoChrome = False
app.Math.XY.PersistenceTime = "Infinite"
```

**AxisXRotation***Integer*

Range: From -90 to 90, step 1

## Description

Sets/Queries the state of the X-axis rotation control, used only in 3-D persistence modes to control the viewing position. See the general description above for a discussion of the locked and unlocked persistence modes.

**AxisYRotation***Integer*

Range: From -90 to 90, step 1

## Description

Sets/Queries the state of the Y-axis rotation control, used only in 3-D persistence modes to control the viewing position.

**PART TWO: REFERENCE**

---

**ClearSweeps****Action**

## Description

Clears the persistence X-Y plot.

**InputX****Enum**

## Description

Sets/Queries the name of the input channel for the X-axis of the X-Y plot.

## Values

BadBits  
Bits  
C1  
C2  
C3  
C4  
Eye  
F1  
F2  
F3  
F4  
F5  
F6  
F7  
F8  
M1  
M2  
M3  
M4  
PRBS

**InputY****Enum**

## Description

Sets/Queries the name of the input channel for the Y-axis of the X-Y plot.

## Values

BadBits  
Bits  
C1  
C2  
C3  
C4  
Eye  
F1  
F2  
F3  
F4  
F5  
F6  
F7

## CHAPTER TWO

## Control Reference

F8  
M1  
M2  
M3  
M4  
PRBS

**Persist3DQuality****Enum**

## Description

Sets/Queries the state of the 3-D Persistence quality control. Controls the way that the persistence trace is rendered.

## Values

Shaded  
Solid  
WireFrame

**Persisted****Bool**

## Description

Sets/Queries the persisted state of the X-Y plot. If the Display.LockPersistence control is set to "AllLocked", the persisted state of all displayed waveforms will be the same. If the Display.LockPersistence control is set to "PerTrace", the persisted state of each waveform can be independently controlled.

**Persistence3d****Bool**

## Description

Sets/Queries the 3-D persistence state. When True, the persistence display for the X-Y plot will be displayed as a three dimensional surface map.

**PersistenceMonoChrome****Bool**

## Description

Sets/Queries the monochrome persistence state. When True, the persistence display for the X-Y plot will be monochromatic, whether 2-D or 3-D.

**PersistenceSaturation****Integer**

Range: From 1 to 100, step 1

## Description

Sets/Queries the saturation threshold for persisted X-Y plot. All information at this level or above will be recorded with the same color or intensity.

**PersistenceTime****Enum**

## Description

Sets/Queries the state of the Persistence Time control. Controls the persistence decay time for the X-Y persistence.

## Values

0.5s  
10s

## PART TWO: REFERENCE

---

1s  
20s  
2s  
5s  
Infinite

### **ShowLastTrace**

***Bool***

#### Description

Sets/Queries the state of the Show Last Trace control. If True, when this trace is displayed in persistence mode, the last acquired waveform will be superimposed on the accumulating persistence map.

### **UseDotJoin**

***Bool***

#### Description

Sets/Queries whether dot joining is used in the X-Y plot.

## RESULT

*app.Math.XY.Out.Result*

Properties of the type `xxxx.Out.Result.yyyy` are those of the last completed acquisition. They are not affected if other CVars are changed after that acquisition was completed. This distinction between "Out.Result" properties and other CVars is most important when the trigger mode is Single or Stopped. You should treat "Out.Result" properties as read-only.

This XY result object is very similar to, but not identical to, the result object exposed by the channel and math traces. The differences are due to the fact that the XY trace returns pairs of data values, one for X, one for Y.

<b>DataArray</b>	Property
<b>FirstEventTime</b>	Property
<b>HorizontalFrameStart</b>	Property
<b>HorizontalFrameStop</b>	Property
<b>HorizontalOffset</b>	Property
<b>HorizontalPerStep</b>	Property
<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>LastEventTime</b>	Property
<b>NumFrameDimensions</b>	Property
<b>Samples</b>	Property
<b>Sweeps</b>	Property
<b>XFrameStart</b>	Property
<b>XFrameStop</b>	Property
<b>XMaxPossible</b>	Property
<b>XMinPossible</b>	Property
<b>XOffset</b>	Property
<b>XPerStep</b>	Property
<b>XResolution</b>	Property
<b>XUnits</b>	Property
<b>YFrameStart</b>	Property
<b>YFrameStop</b>	Property
<b>YMaxPossible</b>	Property
<b>YMinPossible</b>	Property
<b>YOffset</b>	Property
<b>YPerStep</b>	Property
<b>YResolution</b>	Property
<b>YUnits</b>	Property

**DataArray***Property*

## Description

This is the array of data that can be read out to represent the calculated waveform. The returned array will have `numSamples` rows, and two columns. The first column will contain X values, and the second column Y values. For example: (0, 0) indicates the X value of the first sample, (10, 1) indicates the Y value of the 10th sample. If `DataArray(True)` is specified, floating-point values are retrieved. If `DataArray(False)` is specified, 16-bit integer values are returned.

**FirstEventTime***Property*

## Description

## PART TWO: REFERENCE

---

Reads the time of the first event.

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the time of the first event.
EventFirst = app.Math.XY.Out.Result.FirstEventTime
```

### HorizontalFrameStart

*Property*

#### Description

Reads the start of the horizontal scale of the Y-T traces that contribute to the X-Y plot. If the input traces have been shifted to different offsets, this CVar refers to the X-trace. The values for both traces can be obtained using `app.Math.Cx.Out.Result.HorizontalFrameStart`.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the horizontal frame start for the input traces of an X-Y
plot.
HorStart = app.Math.XY.Out.Result.HorizontalFrameStart
```

### HorizontalFrameStop

*Property*

#### Description

Reads the end of the horizontal scale of the Y-T traces that contribute to the X-Y plot. If the input traces have been shifted to different offsets, this control variable refers to the X-trace.

### HorizontalOffset

*Property*

#### Description

Reads the offset of the horizontal scale of the Y-T traces that contribute to the X-Y plot. If the input traces have been shifted to different offsets, this cvar refers to the X-trace. The values for both traces can be obtained using `app.Math.Cx.Out.Result.HorizontalFrameStart`.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the horizontal offset of the input traces of an XY plot.
HorOffset = app.Math.XY.Out.Result.HorizontalOffset
```

### HorizontalPerStep

*Property*

#### Description

This is the time between successive sampling instants of the input traces of an X-Y plot.

#### Example

**CHAPTER TWO****Control Reference**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the value of the horizontal step size for the input traces
of an X-Y plot.
HorStepXY = app.Math.XY.Out.Result.HorizontalPerStep
```

**HorizontalResolution****Property**

## Description

This is the resolution of the readout of horizontal values of the input traces of an X-Y plot. It is not directly related to the sampling period.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the readout resolution of the horizontal axis of input
traces of an X-Y plot.
HRes = app.Math.XY.Out.Result.HorizontalResolution
```

**HorizontalUnits****Property**

## Description

Reads the name of the horizontal units of the data of the X-Y plot. The horizontal dimension is not visible on the X-Y plot, but is implicit in the data. If both inputs are normal channel traces, units are in seconds; while if both are FFTs, units are in Hz.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the horizontal units of the input traces of an XY plot.
HUnits = app.Math.XY.Out.Result.HorizontalUnits
```

**LastEventTime****Property**

## Description

Reads the time of the last event.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the time of the last event.
EventLast = app.Math.XY.Out.Result.LastEventTime
```

**PART TWO: REFERENCE****NumFrameDimensions****Property**

## Description

Reads the dimensionality of trace XY: 2 for a Y-T plot, 3 for an X-Y plot.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the dimensionality of trace XY.
DimXY = Math.XY.Out.Result.NumFrameDimensions
```

**Samples****Property**

## Description

Reads the number of points in the input traces of an X-Y plot.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the number of samples in the input traces of an X-Y plot.
XYSamples = app.Math.XY.Out.Result.Samples
```

**Sweeps****Property**

## Description

This is the number of sweeps since the last clearance, for example a clearance caused by the last change in time per division.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the number of sweeps for trace XY.
XYSweeps = app.Math.XY.Out.Result.Sweeps
```

**XFrameStart****Property**

## Description

Reads the value at the left edge of the X-Y plot.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the value at the left hand edge for the X-Y plot.
XLeft = app.Math.XY.Out.Result.XFrameStart
```

## CHAPTER TWO

## Control Reference

**XFrameStop***Property*

## Description

See the corresponding control variable in **app.Acquisition.Cx.Out.Result**.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the value at the right hand edge for the X-Y plot.
XRight = app.Math.XY.Out.Result.XFrameStop
```

**XMaxPossible***Property*

## Description

This is the highest value that an actual array element can have. It will be a little less than XFrameStop.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the maximum possible value of X for trace XY.
XMaxXY = app.Math.XY.Out.Result.XMaxPossible
```

**XMinPossible***Property*

## Description

This is the lowest value that an actual array element can have. It will be a little greater than XFrameStart.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the minimum possible value of X for trace XY.
XMinXY = app.Math.XY.Out.Result.XMinPossible
```

**XOffset***Property*

## Description

See the corresponding control variable in **app.Acquisition.Cx.Out.Result**.

**XPerStep***Property*

## Description

This is the smallest step in the numerical X values that can be read out, whether or not the step has physical meaning. For the basic 8-bit ADC the step is 1/65536 of the X-range.

## PART TWO: REFERENCE

---

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the step size in X for trace XY.
XStepXY = app.Math.XY.Out.Result.XPerStep
```

### XResolution

### Property

#### Description

X resolution is the actual smallest difference that can be practically resolved. For an 8-bit ADC it is 1/256 of the height of the vertical range. But if 16 averages are set, the resolution is improved by a factor of 4, and it becomes 1/1024 of the vertical range, and for 100 sweeps it becomes 1/2560.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the resolution of the X-axis of trace XY.
XRes = app.Math.XY.Out.Result.Xresolution
```

### XUnits

### Property

#### Description

Reads the name of the units of the horizontal scale of an XY plot. This is not the same as the horizontal unit, `app.Math.XY.Out.Result.HorizontalUnits`, which refers to the horizontal units of the original traces that contribute to the XY plot.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the units of the horizontal axis of an XY plot.
XUnits = app.Math.XY.Out.Result.Xunits
```

### YFrameStart

### Property

#### Description

Reads the value at the bottom edge for the X-Y plot.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the value at the bottom edge for the X-Y plot.
YBottom = app.Math.XY.Out.Result.YFrameStart
```

**CHAPTER TWO****Control Reference**

---

**YFrameStop****Property**

## Description

Reads the value at the top edge for the X-Y plot.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the value at the top edge for the X-Y plot.
YTop = app.Math.XY.Out.Result.YFrameStop
```

**YMaxPossible****Property**

## Description

This is the highest value that an actual array element can have. It will be a little less than YFrameStop.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the maximum possible value of Y for trace XY.
YMaxXY = app.Math.XY.Out.Result.YMaxPossible
```

**YMinPossible****Property**

## Description

This is the lowest value that an actual array element can have. It will be a little greater than YFrameStart.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the minimum possible value of Y for trace XY.
YMinXY = app.Math.XY.Out.Result.YMinPossible
```

**YOffset****Property**

## Description

Please see the corresponding control variable in app.Acquisition.Cx.Out.Result.

**YPerStep****Property**

## Description

This is the smallest step in the numerical Y values that can be read out, whether or not the step has physical meaning. For the basic 8-bit ADC, the step is 1/65536 of the Y-range.

## PART TWO: REFERENCE

---

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the step size in Y for trace XY.
YStepXY = app.Math.XY.Out.Result.YPerStep
```

### YResolution

### *Property*

#### Description

Y-resolution is the actual smallest difference that can be practically resolved. For an 8-bit ADC, it is 1/256 of the height of the Y-range. But if 16 averages are set in the channel, the resolution is improved by a factor of 4, and it becomes 1/1024. For 100 sweeps it becomes 1/2560.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the resolution of the Y-axis of trace XY.
YRes = app.Math.XY.Out.Result.Yresolution
```

### YUnits

### *Property*

#### Description

Reads the name of the units of the vertical scale of an XY plot.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the units of the vertical axis of an XY plot.
YUnits = app.Math.XY.Out.Result.Yunits
```

**MEASURE***app.Measure*

Variables of the form `app.Measure` control the parameters P1 through P8, and their associated statistical results and histicons.

Names of the form `app.Measure.Measure("Premote").xxxx` and `app.Measure.Measure("Px").xxxx` are aliases for simpler names, which are described in this section of the manual. Examples of alias pairs are as follows:

```
app.Measure.Measure("Premote").OutResult = app.Measure."Premote".OutResult
app.Measure.Measure("Px").Statistics = app.Measure.Px.Statistics
```

<b>ClearAll</b>	Action
<b>ClearSweeps</b>	Action
<b>HelpMarkers</b>	Enum
<b>HistoOn</b>	Bool
<b>MeasureMode</b>	Enum
<b>SetGateToDefault</b>	Action
<b>StatsOn</b>	Bool
<b>StdGateStart</b>	Double
<b>StdGateStop</b>	Double
<b>StdSource</b>	Enum

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' get into the custom parameter mode
app.Measure.MeasureMode = "MyMeasure"
app.Measure.ClearAll
app.Measure.StatsOn = True
app.Measure.HistoOn = False

' Configure P1 to measure amplitude
app.Measure.P1.View = True
app.Measure.P1.ParamEngine = "ampl"
```

**ClearAll****Action**

## Description

Resets all parameter setups, turning each of the parameters view to "off", the MeasurementType to "measure" and the selected paramEngine to "Null". See **Acquisition.Channels** for a programming example.

**ClearSweeps****Action**

## Description

Clears the accumulated statistics for parameters P1 to P8, as well as the accumulated statistics for their associated histicons.

## PART TWO: REFERENCE

### HelpMarkers

*Enum*

#### Description

Sets/Queries the level of detail for help markers (if any of the selected parameter definitions have help markers). These markers are displayed on the source traces, but only if those traces are viewed simultaneously with the parameter measurements. **Note:** This setting is global for all Px.

#### Values

<b>Detailed</b>	Detailed help markers
<b>Off</b>	No help markers
<b>Simple</b>	Simple help markers

### HistoOn

*Bool*

#### Description

Sets/Queries the visibility of the histicons of the parameters that are viewed.

### MeasureMode

*Enum*

#### Description

Sets/Queries the set of parameters to be displayed.

#### Values

<b>MyMeasure</b>	Completely customizable user defined parameter list, .
<b>Off</b>	
<b>StdHorizontal</b>	Standard horizontal parameters
<b>StdVertical</b>	Standard vertical parameters

### SetGateToDefault

*Action*

#### Description

Sets the measure gate to its default state. Valid only when in either Std. Vertical or Std. Horizontal measurement modes. For MyMeasure, see the equivalent controls under Px.

### StatsOn

*Bool*

#### Description

Sets/Queries the display of parameter statistics. **Note:** Statistics are accumulated whether or not they are displayed, i.e., you do not need to set StatsOn = "On" to collect statistics.

### StdGateStart

*Double*

Range: From 0 to 10, step 0.01

#### Description

Sets/Queries the position of the left limit of the measure gate (in divisions). Valid only in either Std. Vertical or Std. Horizontal measurement modes. For MyMeasure, see the equivalent controls under Px.

**StdGateStop***Double*

Range: From 0 to 10, step 0.01

## Description

Sets/Queries the position of the right-hand limit of the measure gate (in divisions). Valid only in either Std. Vertical or Std. Horizontal measurement modes. For MyMeasure see the equivalent controls under Px.

**StdSource***Enum*

## Description

Sets/Queries the channel that is the source of ALL standard voltage or time parameters. In “My Measure” mode, each parameter has its own Source selection, and this setting is ignored.

## Values

BadBits  
Bits  
C1  
C2  
C3  
C4  
Ctie  
Eye  
F1  
F2  
F3  
F4  
F5  
F6  
F7  
F8  
M1  
M2  
M3  
M4  
PRBS  
TIE

## PART TWO: REFERENCE

**MEASURE***app.Measure.Measure*

Names of the form `app.Measure.Measure("Premote").xxxx` and `app.Measure.Measure("Px").xxxx` are aliases for simpler names, which are described in the section of the manual devoted to `app.Measure`. Examples of alias pairs are as follows:

```
app.Measure.Measure("Premote").OutResult = app.Measure."Premote".OutResult
app.Measure.Measure("Px").Statistics = app.Measure.Px.Statistics
```

**PX****app.Measure.Px**

This set of variables controls parameters P1 through P8, and the statistical results and histograms that depend on them. Valid only when the `MeasureMode` is "MyMeasure", otherwise they are predefined.

<b>ArithEngine</b>	Enum
<b>Equation</b>	String
<b>GateStart</b>	Double
<b>GateStop</b>	Double
<b>MeasurementType</b>	Enum
<b>ParamEngine</b>	Enum
<b>PSource1</b>	Enum
<b>PSource2</b>	Enum
<b>Source1</b>	Enum
<b>Source2</b>	Enum
<b>View</b>	Bool

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

app.Measure.MeasureMode = "MyMeasure"

' Set parameter P1 to math on parameters.
App.Measure.P1.MeasurementType = "math"
```

**ArithEngine****Enum**

## Description

Sets/Queries the parameter arithmetic function for parameter Px. The setting is only used when `Px.MeasurementType` is set to "math".

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the parameter arithmetic for P1 to product of two
parameters.
app.Measure.MeasureMode = "MyMeasure"
app.Measure.P1.MeasurementType = "math"
```

## CHAPTER TWO

## Control Reference

```
app.Measure.P1.ArithEngine = "ParamProduct"
app.Measure.P1.PSource1 = "P2"
app.Measure.P1.PSource2 = "P3"
```

'of course to get an answer, you should also define P2 and P3

```
app.Measure.P2.MeasurementType = "measure"
app.Measure.P2.ParamEngine = "Ampl"
app.Measure.P2.Source1 = "C1"
```

```
app.Measure.P3.MeasurementType = "measure"
app.Measure.P3.ParamEngine = "Mean"
```

## Values

```
MathcadParamArith
ParamConst
ParamDifference
app.Measure.P3.Source1 = "C2"ParamInvert
ParamProduct
ParamRatio
ParamScript
ParamSum
```

**Equation****String**

Range: Any number of characters.

Description

Reads the text equation that summarizes the setup for parameter Px.

**GateStart****Double**

Range: From 0 to 10, step 0.01

Description

Sets/Reads the position of the left edge of the measure gate for parameter Px.

**GateStop****Double**

Range: From 0 to 10, step 0.01

Description

Sets/Reads the position of the right-hand edge of the measure gate for parameter Px.

**MeasurementType****Enum**

Description

Sets/Queries the measurement type of the parameter Px.

## Values

<b>math</b>	Measurement is a mathematical combination of one or two other Py,Pz
<b>measure</b>	Standard measurement mode (parametric of a trace waveform)
<b>WebEdit</b>	Measurement is defined using the Processing Web Editor

## PART TWO: REFERENCE

---

### ParamEngine

*Enum*

#### Description

Sets/Queries the parameter (measurement on a trace) for Px. This setting applies only if the MeasurementType control is set to "measure".

#### Values

Amplitude  
Area  
Base  
Cycles  
Delay  
DeltaDelay  
DeltaPeriodAtLevel  
DeltaTimeAtLevel  
DeltaWidthAtLevel  
Duration  
DutyAtLevel  
DutyCycle  
EdgeAtLevel  
ExcelParam  
ExtinctionRatio  
EyeAmplitude  
EyeAvgPower  
EyeBER  
eyecrossing  
EyeHeight  
EyeOneLevel  
EyeQ  
EyeWidth  
EyeZeroLevel  
Fall  
Fall8020  
FallAtLevel  
FirstPoint  
Frequency  
FrequencyAtLevel  
FullWidthAtHalfMaximum  
FullWidthAtXX  
HalfPeriod  
HistogramAmplitude  
HistogramBase  
HistogramMaximum  
HistogramMean  
HistogramMedian  
HistogramMinimum  
HistogramRms  
HistogramSdev  
HistogramTop  
HoldTime  
LastPoint  
LevelAtX  
MathcadParam

---

MATLABParameter  
 Maximum  
 MaximumPopulation  
 Mean  
 Median  
 Minimum  
 Mode  
 NarrowBandPhase  
 NarrowBandPower  
 npoints  
 Null  
 NumberOfModes  
 OvershootNegative  
 OvershootPositive  
 ParamScript  
 Peaks  
 PeakToPeak  
 Percentile  
 Period  
 PeriodAtLevel  
 Phase  
 PopulationAtX  
 Range:  
 Rise  
 Rise2080  
 RiseAtLevel  
 RootMeanSquare  
 Setup  
 Skew  
 StandardDeviation  
 TIE  
 TimeAtLevel  
 Top  
 TotalPopulation  
 Width  
 WidthAtLevel  
 XAtMaximum  
 XAtMinimum  
 XAtPeak

**PSource1****Enum**

## Description

Sets/Queries the first parameter source of parameter Px. This choice is only applicable if the MeasurementType is set to "math". The choices are limited to other parameters.

## Values

P1  
 P2  
 P3  
 P4  
 P5  
 P6

## PART TWO: REFERENCE

---

P7

P8

### **PSource2**

*Enum*

#### Description

Sets/Queries the second parameter source of the parameter Px, when the MeasurementType is "math".

#### Values

P1

P2

P3

P4

P5

P6

P7

P8

### **Source1**

*Enum*

#### Description

Sets/Queries the first trace source of the parameter Px. Used only when MeasurementType = "measure". For MeasurementType = "math", refer to PSource1.

#### Values

C1

C2

C3

C4

F1

F2

F3

F4

F5

F6

F7

F8

M1

M2

M3

M4

### **Source2**

*Enum*

#### Description

Sets/Queries the second trace source of the parameter Px. Only applicable when the MeasurementType is "measure". The choices are limited to waveforms.

#### Values

C1

C2

C3

C4

## CHAPTER TWO

Control Reference

---

F1  
F2  
F3  
F4  
F5  
F6  
F7  
F8  
M1  
M2  
M3  
M4  
None

**View****Bool**

## Description

Sets/Queries the display of parameter Px.

**Note:** It is not necessary for a parameter to be displayed in order for it to be used in combinatorial math (i.e., as a source for another parameter in MeasurementType "math").

**PART TWO: REFERENCE**

---

**OPERATOR**

*app.Measure.Px.Operator*

This path specifies that the selected ParamEngine or ArithEngine control variables are "here."

**RESULT****app.Measure.Px.Out.Result**

Properties of the type `xxxx.Out.Result.yyyy` are those of the last completed acquisition. They are not affected if other CVars are changed after that acquisition was completed. This distinction between "Out.Result" properties and other CVars is most important when the trigger mode is Single or Stopped. You should treat "Out.Result" properties as read-only.

<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>Value</b>	Property
<b>VerticalResolution</b>	Property
<b>VerticalUnits</b>	Property

**HorizontalResolution****Property**

## Description

Reads the readout resolution of horizontal values. This is not directly related to the sample rate.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the readout resolution of the horizontal axis of the trace
' on which the parameter P4 is based.
HRes = app.Measure.P4.Out.Result.HorizontalResolution
```

**HorizontalUnits****Property**

## Description

Reads the horizontal unit of the trace on which the parameter Px is based.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal unit for parameter P1.
HorUnit = app.Measure.P1.Out.Result.HorizontalUnits
```

**Value****Property**

## Description

Reads the value of the parameter.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the value of the parameter P3.
P3Value = app.Measure.P3.Out.Result.Value
```

## PART TWO: REFERENCE

---

### VerticalResolution

### *Property*

#### Description

Vertical resolution is the actual smallest difference that can be practically resolved. For an 8-bit ADC, it is 1/256 of the height of the vertical range (VR). But if 16 averages are set in the channel, the resolution is improved by a factor of 4, and it becomes 1/1024 of VR. For 100 sweeps it becomes 1/2560.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the readout resolution of the vertical axis of the trace
' on which the parameter P4 is based.
VRes = app.Measure.P4.Out.Result.VerticalResolution
```

### VerticalUnits

### *Property*

#### Description

Reads the vertical unit of the trace on which parameter Px is based.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical unit for parameter P1.
VerUnit = app.Measure.P1.Out.Result.VerticalUnits
```

**STATISTICS***app.Measure.Px.Statistics*

This set of variables controls the statistical summaries that are provided for all the parameters.

**RESULT***app.Measure.Px.Statistics("histo").Result*

This set of variables provides data from the histicons that are provided for all the parameters.

<b>Base</b>	Property
<b>BinPopulations</b>	Property
<b>Bins</b>	Property
<b>BinWidth</b>	Property
<b>FirstPopulatedBin</b>	Property
<b>HorizontalFrameStart</b>	Property
<b>HorizontalFrameStop</b>	Property
<b>HorizontalUnits</b>	Property
<b>LastPopulatedBin</b>	Property
<b>Max</b>	Property
<b>MaxPopulation</b>	Property
<b>MaxPopulationBin</b>	Property
<b>Mean</b>	Property
<b>Min</b>	Property
<b>OffsetAtLeftEdge</b>	Property
<b>PeakInfo</b>	Property
<b>Peaks</b>	Property
<b>PopulationInside</b>	Property
<b>PopulationOver</b>	Property
<b>PopulationUnder</b>	Property
<b>Rms</b>	Property
<b>Sdev</b>	Property
<b>Sweeps</b>	Property
<b>Top</b>	Property
<b>VerticalFrameStart</b>	Property
<b>VerticalFrameStop</b>	Property
<b>VerticalUnits</b>	Property

**Base****Property**

## Description

Reads the base value of the distribution.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
app.Measure.P1.ParamEngine = "StandardDeviation"
' Read the base value in the histicon for parameter P1.
baseValue = app.Measure.P1.Statistics("histo").Result.Base
```

## PART TWO: REFERENCE

---

### BinPopulations

*Property*

#### Description

Reads the array of bin populations for the distribution.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
' Collect the number of bins in the histicon of parameter P1.
Bins = app.Measure.P1.Statistics("histo").Result.Bins
' Collect the array of bin populations for the histogram of
parameter P1.
dim BinPop(100)
for i = 0 to 100
BinPop(i) =
app.Measure.P1.Statistics("histo").Result.BinPopulations(i)
next
```

### Bins

*Property*

#### Description

Reads the number of bins in the histogram.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"

' Read the number of bins in the histogram for parameter P1.
numberOfBins = app.Measure.P1.Statistics("histo").Result.Bins
```

### BinWidth

*Property*

#### Description

Reads the width of the bins of the histogram.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure

' Inspect the histicon bin width for parameter P8.
HB8Width = app.Measure.P8.Statistics("histo").BinWidth
```

## CHAPTER TWO

## Control Reference

**FirstPopulatedBin***Property*

## Description

Reads the position of the leftmost populated bin of the histogram.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
' Read the position of the leftmost populated bin
' of the histogram for parameter P1.
BinLeftP1 =
app.Measure.P1.Statistics("histo").Result.FirstPopulatedBin
```

**HorizontalFrameStart***Property*

## Description

Queries the left edge of the frame of the histogram for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal frame start of the histogram for
parameter P3.
LeftEdge =
app.Measure.P3.Statistics("histo").Result.HorizontalFrameStart
```

**HorizontalFrameStop***Property*

## Description

Reads the right-hand edge of the frame of the histogram of parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal frame stop of the histogram for parameter
P3.
RightEdge =
app.Measure.P3.Statistics("histo").Result.HorizontalFrameStop
```

**HorizontalUnits***Property*

## Description

Reads the horizontal dimensional or physical units for the histogram. **Note:** These are nominally identical to the physical units of the parametric measurement that has been histogrammed.

**PART TWO: REFERENCE**

---

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
' Inspect the horizontal unit for the histicon for parameter P1.
HorUnit =
```

**LastPopulatedBin****Property****Description**

Reads the index of the rightmost populated bin of the histogram.

**Max****Property****Description**

Reads the maximum value for the histicon of parameter Px.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
' Read the maximum value for the histicon for parameter P1.
coordinateOfRightmostPopulated =
app.Measure.P1.Statistics("histo").Max
```

**MaxPopulation****Property****Description**

Reads the population of the most populated bin of the histogram.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the population of the most populated bin
' of the histicon for parameter P6.
BinMaxHP6 =
```

**MaxPopulationBin****Property****Description**

Reads the index of the most populated bin of the histogram.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

**CHAPTER TWO****Control Reference**

```
' Read the index of the bin with the maximum population
indexOfMaxPopulation =
app.Measure.P1.Statistics("histo").Result.MaxPopulationBin
```

**Mean****Property**

## Description

Reads the mean of the distribution of the histogram.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
' Inspect the mean of the histogram for the parameter P1.
mean = app.Measure.P1.Statistics("histo").Mean
```

**Min****Property**

## Description

Reads the minimum value of the histogram.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
' Read the maximum value for the histogram for parameter P1.
coordinateOfLeftmostPopulated =
app.Measure.P1.Statistics("histo").Result.Min
```

**OffsetAtLeftEdge****Property**

## Description

Reads the horizontal coordinate of the left edge of the leftmost bin in the histogram.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Look at the left edge offset of the histogram for parameter P1.
leftEdge =
app.Measure.P1.Statistics("histo").Result.OffsetAtLeftEdge
```

**PeakInfo****Property**

## Description

Reads three values about a peak of the histogram of parameter Px. The first two results are the mean and the standard deviation. The third value is unused.

## PART TWO: REFERENCE

---

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Collects the information array for second peak of
' the histicon of parameter P3.
ReDim InfoArray2
InfoArray2 = app.Measure.P3.Statistics("histo").Result.PeakInfo(2)
```

### Peaks

#### *Property*

#### Description

Reads the number of peaks in the distribution (according to a peak identification process).

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the number of peaks found
numPeaks = app.Measure.P1.Statistics("histo").Result.Peaks
```

### PopulationInside

#### *Property*

#### Description

Reads the total number of data points inside the frame of the histicon of parameter Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the population inside the histicon for parameter P3.
PopInP3 = app.Measure.P3.Statistics("histo").Result.PopulationInside
```

### PopulationOver

#### *Property*

#### Description

Reads the number of data points outside the histicon to the right of the last bin in the histogram

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the population to the right of the frame
' of the histicon for parameter P1.
populationInOverflow =
app.Measure.P1.Statistics("histo").Result.PopulationOver
```

**CHAPTER TWO****Control Reference****PopulationUnder****Property**

## Description

Reads the number of data points outside the histogram to the left of the leftmost bin.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
' Read the population that are NOT binned because the values were
  to the left of the leftmost bin
numUnderflows =
app.Measure.P1.Statistics("histo").Result.PopulationUnder
```

**Rms****Property**

## Description

Reads the root mean square of the distribution.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
' Read the rms value of the histicon for parameter P1.
rms = app.Measure.P1.Statistics("histo").Rms
```

**Sdev****Property**

## Description

Reads the standard deviation of the distribution.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the standard deviation of the histogrammed distribution
standardDeviation = app.Measure.P1.Statistics("histo").Sdev
```

**Sweeps****Property**

## Description

Reads the number of sweeps contributing to the histogram.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the number of sweeps for the histicon for parameter P1.
```

**PART TWO: REFERENCE**

---

**Top****Property**

## Description

Reads the top value of the histicon for the parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the top value in the histicon for parameter P6.
TopHP6 = app.Measure.P6.Statistics("histo").Top
```

**VerticalFrameStart****Property**

## Description

Reads the value at the bottom of the frame (display graticule)of the histogram.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical frame start of the histicon for parameter P1.
BottomEdge = app.Measure.P1.Statistics("histo").VerticalFrameStart
```

**VerticalFrameStop****Property**

## Description

Reads the value at the top of the frame of the histogram.

**VerticalUnits****Property**

## Description

Reads the vertical units for the histogram. These are almost always "Events."

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical units of the histicon of parameter P4.
UVert = app.Measure.P4.Statistics("histo").Result.VerticalUnits
```

**RESULT***app.Measure.Px.Statistics("last").Result*

This set of variables controls the readout of data from the last event in the statistics that are provided for all the parameters.

<b>Value</b>	Property
<b>ValueArray</b>	Property
<b>VerticalResolution</b>	Property
<b>VerticalUnits</b>	Property

**Value***Property*

## Description

Reads the last recorded value of parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the last recorded value for the parameter P3.
LastValue = app.Measure.P3.Statistics("last").Result.Value
```

**ValueArray***Property*

## Description

Some measurements return multiple values for a single waveform input. This array contains all of the values.

**VerticalResolution***Property*

## Description

Reads the vertical resolution of the result.

**VerticalUnits***Property*

## Description

Reads the vertical unit of measurement for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical unit of measurement for the parameter P1.
VerUnit = app.Measure.P1.Statistics("last").Result.VerticalUnits
```

## PART TWO: REFERENCE

---

### RESULT

### *app.Measure.Px.Statistics("max").Result*

This set of variables controls the readout of data about the maximum value in the statistics that are provided for all the parameters.

<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>Value</b>	Property
<b>VerticalResolution</b>	Property
<b>VerticalUnits</b>	Property

### HorizontalResolution

### *Property*

#### Description

Reads the horizontal resolution for the parameter Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal resolution for the parameter P1.
HorRes = app.Measure.P1.Statistics("max").Result.HorizontalResolution
```

### HorizontalUnits

### *Property*

#### Description

Reads the horizontal units of measurement for parameter Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal unit for the parameter P1.
HorUnit = app.Measure.P1.Statistics("max").Result.HorizontalUnits
```

### Value

### *Property*

#### Description

Reads the maximum recorded value of parameter Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the highest recorded value for the parameter P5.
MaxValue = app.Measure.P5.Statistics("max").Result.Value
```

**CHAPTER TWO****Control Reference**

---

**VerticalResolution***Property*

## Description

Reads the vertical resolution for the parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical resolution for the parameter P1.
VerRes = app.Measure.P1.Statistics("max").Result.VerticalResolution
```

**VerticalUnits***Property*

## Description

Reads the vertical unit of measurement for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical unit for the parameter P5.
VerUnit = app.Measure.P5.Statistics("max").Result.VerticalUnits
```

## PART TWO: REFERENCE

**RESULT****app.Measure.Px.Statistics("mean").Result**

This set of variables controls the readout of data about the mean value in the statistics that are provided for all the parameters.

**Result**

<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>Value</b>	Property
<b>VerticalResolution</b>	Property
<b>VerticalUnits</b>	Property

**HorizontalResolution****Property**

## Description

Reads the horizontal resolution for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal resolution for the parameter P1.
HorRes =
app.Measure.P1.Statistics("mean").Result.HorizontalResolution
```

**HorizontalUnits****Property**

## Description

Reads the horizontal units of measurement for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal unit for the parameter P1.
HorUnit = app.Measure.P1.Statistics("mean").Result.HorizontalUnits
```

**Value****Property**

## Description

Reads the mean value of parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the mean recorded value for the parameter P3.
MeanValue = app.Measure.P3.Statistics("mean").Result.Value
```

**CHAPTER TWO****Control Reference**

---

**VerticalResolution*****Property***

## Description

Reads the vertical resolution for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical resolution for the parameter P1.
VerRes = app.Measure.P1.Statistics("mean").Result.VerticalResolution
```

**VerticalUnits*****Property***

## Description

Reads the vertical unit of measurement for the parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical uni for the parameter P1.
VerUnit = app.Measure.P1.Statistics("mean").Result.VerticalUnits
```

## PART TWO: REFERENCE

---

### RESULT

*app.Measure.Px.Statistics("min").Result*

This set of variables controls the readout of data about the minimum value in the statistics that are provided for all the parameters.

<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>Value</b>	Property
<b>VerticalResolution</b>	Property
<b>VerticalUnits</b>	Property

### HorizontalResolution

*Property*

#### Description

Reads the horizontal resolution for parameter Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal resolution for the parameter P1.
HorRes = app.Measure.P1.Statistics("min").Result.HorizontalResolution
```

### HorizontalUnits

*Property*

#### Description

Reads the horizontal units of measurement for parameter Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal unit for the parameter P1.
HorUnit = app.Measure.P1.Statistics("min").Result.HorizontalUnits
```

### Value

*Property*

#### Description

Reads the lowest recorded value of parameter Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the minimum recorded value for the parameter P3.
MinValue = app.Measure.P3.Statistics("min").Result.Value
```

**CHAPTER TWO****Control Reference**

---

**VerticalResolution*****Property***

## Description

Reads the vertical resolution for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical resolution for the parameter P1.
VerRes = app.Measure.P1.Statistics("min").Result.VerticalResolution
```

**VerticalUnits*****Property***

## Description

Reads the vertical unit of measurement for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical unit for the parameter P1.
VerUnit = app.Measure.P1.Statistics("min").Result.VerticalUnits
```

## PART TWO: REFERENCE

---

### RESULT

*app.Measure.Px.Statistics("num").Result*

This set of variables controls the readout of data about the number of data values in the statistics that are provided for all the parameters.

<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>Value</b>	Property
<b>VerticalResolution</b>	Property
<b>VerticalUnits</b>	Property

### HorizontalResolution

*Property*

#### Description

Reads the horizontal resolution for parameter Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal resolution for the parameter P1.
HorRes = app.Measure.P1.Statistics("num").Result.HorizontalResolution
```

### HorizontalUnits

*Property*

#### Description

Reads the horizontal units of measurement for parameter Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal unit for the parameter P1.
HorUnit = app.Measure.P1.Statistics("num").Result.HorizontalUnits
```

### Value

*Property*

#### Description

Reads the number of recorded values of parameter Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the number of values of the parameter P3.
NumValues = app.Measure.P3.Statistics("num").Result.Value
```

**CHAPTER TWO****Control Reference**

---

**VerticalResolution*****Property***

## Description

Reads the vertical resolution for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical resolution for the parameter P1.
VerRes = app.Measure.P1.Statistics("num").Result.VerticalResolution
```

**VerticalUnits*****Property***

## Description

Reads the vertical unit of measurement parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical unit for the parameter P1.
VerUnit = app.Measure.P1.Statistics("num").Result.VerticalUnits
```

## PART TWO: REFERENCE

**RESULT****app.Measure.Px.Statistics("sdev").Result**

This set of variables controls the readout of data about the standard deviation of the values in the statistics that are provided for all the parameters.

**Result**

<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>Value</b>	Property
<b>VerticalResolution</b>	Property
<b>VerticalUnits</b>	Property

**HorizontalResolution****Property**

## Description

Reads the horizontal resolution for the parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal resolution for the parameter P1.
HorRes =
app.Measure.P1.Statistics("sdev").Result.HorizontalResolution
```

**HorizontalUnits****Property**

## Description

Reads the horizontal units of measure for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal unit for the parameter P1.
HorUnit = app.Measure.P1.Statistics("sdev").Result.HorizontalUnits
```

**Value****Property**

## Description

Reads the standard deviation of the recorded values of parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the standard deviation of therecorded values
' of the parameter P5.
```

---

```
SDevValue = app.Measure.P5.Statistics("sdev").Result.Value
```

**VerticalResolution****Property**

## Description

Reads the vertical resolution for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical resolution for the parameter P1.
VerRes = app.Measure.P1.Statistics("sdev").Result.VerticalResolution
```

**VerticalUnits****Property**

## Description

Reads the vertical units for parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical unit for the parameter P1.
VerUnit = app.Measure.P1.Statistics("sdev").Result.VerticalUnits
```

## PART TWO: REFERENCE

---

### MEMORY

#### *app.Memory*

Variables of the form `app.Memory.xxxx` control the memories (M1 to M4). Names of the form `app.Memory.Memories("Mx").xxx` are aliases for simpler names, which are described in this section of the manual. Examples of alias pairs are as follows:

```
app.Memory.Memories("Mx").Out.Result = app.Memory.Mx.Out.Result
```

```
app.Memory.Memories("Mx").Zoom = app.Memory.Mx.Zoom
```

See `app.Acquisition.Channels("Cx")` for a programming example.

#### **ClearAllMem**

#### **Action**

##### Description

Clears the contents of all trace memories.

##### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Clear the contents of all trace memories.
app.Memory.ClearAllMem
```

---

**MEMORIES***app.Memory.Memories*

Names of the form `app.Memory.Memories("Mx").xxxx` are aliases for simpler names, which are described in the section of the manual devoted to `app.Memory`. Examples of alias pairs are as follows:

`app.Memory.Memories("Mx").Out.Result = app.Memory.Mx.Out.Result`

`app.Memory.Memories("Mx").Zoom = app.Memory.Mx.Zoom`

See under **Acquisition.Channels** for a programming example.

## PART TWO: REFERENCE

## MX

*app.Memory.Mx*

This set of variables controls the memories (M1 to M4).

<b>ClearMem</b>	Action
<b>Copy</b>	Action
<b>IncludedInMZgroup</b>	Bool
<b>LabelsPosition</b>	String
<b>LabelsText</b>	String
<b>Source1</b>	Enum
<b>SourceTimeText</b>	String
<b>UseDotJoin</b>	Bool
<b>UseGrid</b>	String
<b>UserText</b>	String
<b>View</b>	Bool
<b>ViewLabels</b>	Bool

**ClearMem****Action**

Description

Initiates a clear memory operation for memory Mx.

**Copy****Action****IncludedInMZgroup****Bool**

Description

Sets/Queries the inclusion of trace MZ in the group to be multi-zoomed.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Include M2 in the multi-zoom group.
app.Memory.M2.IncludedInMZgroup = True
```

**LabelsPosition****String**

Range: Any number of characters

**LabelsText****String**

Range: Any number of characters.

**Source1****Enum**

Values

```
BadBits
Bits
C1
C2
C3
C4
```

## CHAPTER TWO

## Control Reference

Eye  
F1  
F2  
F3  
F4  
F5  
F6  
F7  
F8  
M2  
M3  
M4  
PRBS

**SourceTimeText****String**

Range: Any number of characters.

**UseDotJoin****Bool**

Description

Sets/Queries the style of memory trace Mx.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set memory trace M2 to dot joining off.
app.Memory.M2.UseDotJoin = False
```

**UseGrid****String**

Range: Any number of characters.

Description

Sets/Queries the grid used for memory trace Mx.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set memory trace M2 to use grid YT3.
app.Memory.M2.UseGrid = "YT3"
```

**UserText****String**

Range: Any number of characters

**View****Bool**

Description

Sets/Queries whether memory trace Mx is visible.

## PART TWO: REFERENCE

---

**ViewLabels**

*Bool*

Description

Sets/Queries whether labels are visible for trace Mx.

**RESULT**

*app.Memory.Mx.Out.Result*

See **app.Acquisition.Cx.Out.Result** for a definition of methods and properties used to access the Mx waveform result.

**ZOOM***app.Memory.Mx.Zoom*

This set of variables controls zooming of the memory traces (M1 to M4).

<b>HorPos</b>	Double
<b>HorZoom</b>	Double
<b>ResetZoom</b>	Action
<b>VariableHorZoom</b>	Bool
<b>VariableVerZoom</b>	Bool
<b>VerPos</b>	Double
<b>VerZoom</b>	Double

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Save C1 into M1
app.SaveRecall.Waveform.SaveTo = "Memory"
app.SaveRecall.Waveform.SaveSource = "C1"
app.SaveRecall.Waveform.SaveDestination = "M1"
app.SaveRecall.Waveform.DoSave

' Zoom M1 by x10 Horizontally, and x2 Vertically
app.Memory.M1.Zoom.ResetZoom
app.Memory.M1.Zoom.HorZoom = 10.0
app.Memory.M1.Zoom.HorPos = 0.0
app.Memory.M1.Zoom.VerZoom = 2.0
app.Memory.M1.Zoom.VerPos = 0.0
```

**HorPos****Double**

Range: From -0.5 to 0.5, step (8 digits)

## Description

Sets/Queries the horizontal position of the center of the grid on the zoomed trace Mx. The unit of measure is the screen width, that is, 0.3 means a shift of three of the ten divisions. A positive value moves the trace to the left.

**HorZoom****Double**

Range: From 0.1 to 100000, step (8 digits)

## Description

Sets/Queries the horizontal magnification of trace Mx. The magnification will be in a 1-2-5-10 sequence unless variable horizontal magnification has been set.

**PART TWO: REFERENCE**

---

**ResetZoom****Action**

## Description

Resets the zoom for trace Mx.

**VariableHorZoom****Bool**

## Description

Sets/Queries the ability to zoom horizontally by a continuously variable factor as opposed to a factor that follows a 1-2-5 sequence.

**VariableVerZoom****Bool**

## Description

Sets/Queries the ability to zoom vertically by a continuously variable factor as opposed to a factor that follows a 1-2-5 sequence.

**VerPos****Double**

Range: From -1.5 to 1.5, step (8 digits)

## Description

Sets/Queries the vertical position of the center of the grid on zoomed trace Mx. The unit of measure is the screen height, that is, 0.375 means a shift of three of the eight divisions. A positive value moves the trace downwards.

**VerZoom****Double**

Range: From 0.1 to 256, step (8 digits)

## Description

Sets/Queries the vertical magnification of trace Mx. The magnification will be in a 1-2-5-10 sequence unless variable vertical magnification has been set.

**PASSFAIL***app.PassFail*

Names of the form `app.PassFail("Qremote").xxxx` and `app.PassFail("Qx").xxxx` are aliases for simpler names, which are described in this section of the manual. Examples of alias pairs are as follows:

```
app.PassFail.PassFail("Qremote").Operator = app.PassFail.Qremote.Operator
```

```
app.PassFail.PassFail("Qx").Out.Result = app.PassFail.Qx.Out.Result
```

<b>ActionOn</b>	Enum
<b>Alarm</b>	Bool
<b>EnableActions</b>	Bool
<b>ExecuteScript</b>	Bool
<b>PredefinedConditions</b>	Enum
<b>PrintScreen</b>	Bool
<b>Pulse</b>	Bool
<b>Save</b>	Bool
<b>Stop</b>	Bool
<b>StopAfter</b>	Integer
<b>StopTesting</b>	Bool
<b>Testing</b>	Bool

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Setup Parameter P1 to be the amplitude of C1
app.Measure.MeasureMode = "MyMeasure"
app.Measure.P1.ParamEngine = "Ampl"
app.Measure.P1.Source1 = "C1"
app.Measure.P1.View = True

' Setup Pass/Fail condition Q1 to be a parameter comparison of P1

' condition is false when Ampl(C1) < 100mV
app.PassFail.Q1.ConditionEngine = "ParameterCompare"
app.PassFail.Q1.Present = True
app.PassFail.Q1.Source1 = "P1"
app.PassFail.Q1.Operator.Limit = 0.1
app.PassFail.Q1.Operator.Condition = "Less"
app.PassFail.Q1.View = True

' Enable the Alarm action, enable actions, and start testing
app.PassFail.EnableActions = True
app.PassFail.ActionOn = "Fail"
```

## PART TWO: REFERENCE

---

```
app.PassFail.Alarm = True
```

```
app.PassFail.PredefinedConditions = "AnyTrue"
```

### ActionOn

**Enum**

#### Description

Sets/Queries whether a Pass condition or a Fail condition will initiate the pre-selected actions.

See **app.Acquisition.Channels("Cx")** for a programming example.

#### Values

Fail  
Pass

### Alarm

**Bool**

#### Description

Sets/Queries whether Alarm is included in the PassFail actions.

### EnableActions

**Bool**

#### Description

Sets/Queries whether the selected actions will be executed if the selected PassFail condition is met.

### ExecuteScript

**Bool**

#### Description

Sets/Queries whether ExecuteScript is included in the PassFail actions.

### PredefinedConditions

**Enum**

#### Description

Sets/Queries the logical criteria that must be met in a pass-fail test. For example, the condition "AnyTrue" means that the pass-fail criterion is met if at least one of the test conditions results in a True result.

#### Values

AllFalse  
AllQ1ToQ4OrAllQ5ToQ8  
AllTrue  
AnyFalse  
AnyQ1ToQ4AndAnyQ5ToQ8  
AnyTrue

### PrintScreen

**Bool**

#### Description

Sets/Queries whether Print Screen is included in the PassFail actions.

### Pulse

**Bool**

#### Description

Sets/Queries whether Pulse is included in the PassFail actions. This action emits a pulse from the Aux output socket.

**CHAPTER TWO****Control Reference**

---

**Save****Bool**

## Description

Sets/Queries whether Save is included in the PassFail actions.

**Stop****Bool**

## Description

Sets/Queries whether Stop is included in the PassFail actions.

**StopAfter****Integer**

Range: From 1 to 1000000000, step 1

## Description

Sets/Queries the maximum number of sweeps that will be acquired before testing is halted.

**StopTesting****Bool**

## Description

If Enabled, testing will stop after a number of sweeps defined by the StopAfter control.

**Testing****Bool**

## Description

Sets/Queries whether PassFail testing is on.

**PART TWO: REFERENCE****QX****app.PassFail.Qx**

This set of variables controls the tests Q1 through Q8 in the pass fail system.

<b>ConditionEngine</b>	Enum
<b>Equation</b>	String
<b>Source1</b>	Enum
<b>Source2</b>	Enum
<b>View</b>	Bool

**ConditionEngine****Enum**

## Description

Sets/Queries whether pass-fail test Qx uses mask testing or parameter comparison.

## Values

MaskTestCondition  
ParameterCompare

**Equation****String**

Range: Any number of characters

## Description

Inspects the equation for pass-fail test Qx. A typical equation would be "All P3 < 0.7071".

**Source1****Enum**

## Description

Sets/Queries the first (parameter) source of pass-fail test Qx.

## Values

P1  
P2  
P3  
P4  
P5  
P6  
P7  
P8

**Source2****Enum**

## Description

Sets/Queries the second (waveform) source of pass-fail test Qx.

## Values

C1  
C2  
C3  
C4  
F1  
F2

**CHAPTER TWO****Control Reference**

---

F3  
F4  
F5  
F6  
F7  
F8  
M1  
M2  
M3  
M4

**View****Bool**

## Description

Sets/Queries whether pass-fail test Qx is visible.

**PART TWO: REFERENCE**

---

**RESULT***app.PassFail.Qx.Out.Result*

Properties of the type xxxx.Out.Result.yyyy are those of the last completed acquisition. They are not affected if other CVars are changed after that acquisition was completed. This distinction between "Out.Result" properties and other CVars is most important when the trigger mode is Single or Stopped. You should treat "Out.Result" properties as read-only.

**Result**

<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>Status</b>	Property
<b>StatusDescription</b>	Property
<b>Value</b>	Property
<b>ValueArray</b>	Property

**HorizontalResolution***Property*

## Description

Reads the readout resolution of horizontal values.

**HorizontalUnits***Property*

## Description

Reads the horizontal unit of the horizontal axis.

**Status***Property***StatusDescription***Property***Value***Property*

## Description

Reads the result of test Qx in the pass-fail system.

**ValueArray***Property*

**PREFERENCES***app.Preferences*

This set of variables controls user preferences for the instrument setup and operation.

**Preferences**

<b>AudibleFeedback</b>	Bool
<b>OffsetControl</b>	Enum
<b>Performance</b>	Enum

**AudibleFeedback*****Bool***

## Description

Sets/Queries whether audible feedback is enabled, to sound when a control is touched.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Turn on the audible feedback function.
app.Preferences.AudibleFeedback = True
```

**OffsetControl*****Enum***

## Description

Sets/Queries whether the Vertical Offset constant is in Volts or Divisions when the vertical scale control is changed.

## Values

Div  
Volts

**Performance*****Enum***

## Description

Sets/Queries the variable value that controls the Optimization of the instrument in terms of analysis or display. When set to "Analysis" the display is given low priority and will update less frequently. Use this mode where analysis performance is much more important than display rate.

## Values

Analysis  
AnalysisMid  
Default  
Display  
DisplayMid

## PART TWO: REFERENCE

## EMAIL

*app.Preferences.Email*

This set of variables controls user preferences for the instrument e-mail system. E-mail can be sent when the hardcopy button is pressed, with the hardcopy system appropriately configured. Two standards are supported, SMTP (Simple Mail Transport Protocol), and MAPI (Messaging Application Programming).

<b>DefaultRecipient</b>	String
<b>Mode</b>	Enum
<b>OriginatorAddress</b>	String
<b>SendTestMail</b>	Action
<b>SMTPServer</b>	String

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure the originator and recipient addresses, replace these
with
' appropriate values for your corporate network.
app.Preferences.Email.DefaultRecipient =
"recipientAddress@domain.com"
app.Preferences.Email.Mode = "SMTP"
app.Preferences.Email.OriginatorAddress = "myScope@myDomain.com"
app.Preferences.Email.SMTPServer =
"companySMTPServer.companydomain.com"

' Send a simple test mail
app.Preferences.Email.SendTestMail
```

**DefaultRecipient****String**

Range: Any number of characters

Description

Sets/Queries the default recipient of e-mail transmissions.

**Mode****Enum**

Description

Sets/Queries the transmission mode for e-mail.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set e-Mail mode to MAPI.
app.Preferences.Email.Mode = "MAPI"
```

## CHAPTER TWO

## Control Reference

Values

<b>MAPI</b>	Messaging Application Programming Interface (Uses Outlook Express by default)
<b>SMTP</b>	Simple Mail Transfer Protocol, requires an SMTP server

**OriginatorAddress****String**

Range: Any number of characters

Description

Sets/Queries the originator address for e-mail. This can be any address, and will be used when the recipient replies to an e-mail. The instrument doesn't necessarily have to have its own E-Mail account in order to use this.

**SendTestMail****Action**

Description

Sends a message by e-mail to test the system.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Send an e-Mail message to test the system.
app.Preferences.EMail.SendTestMail
```

**SMTPServer****String**

Range: Any number of characters.

Description

Sets/Queries the name of the SMTP Server for e-mail. Ask your system administrator if you are unsure of what value to set this to.

PART TWO: REFERENCE

---

**SAVERECALL**

*app.SaveRecall*

Controls for the Save/Recall subsystem. Includes nodes for saving and recalling Waveforms and Panels (Setups).

**SETUP***app.SaveRecall.Setup*

These are controls for Saving and Recalling instrument setups.

<b>DoRecallDefaultNvIPanel</b>	Action
<b>DoRecallDefaultPanel</b>	Action
<b>DoRecallPanel</b>	Action
<b>DoSavePanel</b>	Action
<b>InternalName1</b>	String
<b>InternalName2</b>	String
<b>InternalName3</b>	String
<b>InternalName4</b>	String
<b>InternalName5</b>	String
<b>InternalName6</b>	String
<b>PanelFilename</b>	FileName
<b>RecallInternal1</b>	Action
<b>RecallInternal2</b>	Action
<b>RecallInternal3</b>	Action
<b>RecallInternal4</b>	Action
<b>RecallInternal5</b>	Action
<b>RecallInternal6</b>	Action
<b>SaveInternal1</b>	Action
<b>SaveInternal2</b>	Action
<b>SaveInternal3</b>	Action
<b>SaveInternal4</b>	Action
<b>SaveInternal5</b>	Action
<b>SaveInternal6</b>	Action

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Reset to default setup
app.SaveRecall.Setup.DoRecallDefaultPanel

' Store the current setup into the first of the 6 setup stores.
app.SaveRecall.Setup.InternalName1 = "My Setup1"
app.SaveRecall.Setup.SaveInternal1
```

**DoRecallDefaultNvIPanel****Action****Description**

Recalls the factory set NVL (preference) panel settings. These are controls that are not affected when the default panel is recalled, and includes items such as the color preferences, remote control preferences, etc. **Caution:** Use with care, especially when invoking by means of the VBS? remote command via GPIB or TCP/IP, which could result in the controller being disconnected when the default port is selected.

## PART TWO: REFERENCE

---

### Example

```
' Visual Basic Script
  Set app = CreateObject("LeCroy.XStreamDSO")

  ' Recall the factory default nvl panel settings.
  app.SaveRecall.Setup.DoRecallDefaultNvlPanel
```

### DoRecallDefaultPanel

### *Action*

#### Description

Recalls the factory set panel settings.

#### Example

```
' Visual Basic Script
  Set app = CreateObject("LeCroy.XStreamDSO")

  ' Recall the factory default panel settings.
  app.SaveRecall.Setup.DoRecallDefaultPanel
```

### DoRecallPanel

### *Action*

#### Description

Recall the panel file named in the PanelFilename control.

#### Example

```
' Visual Basic Script
  Set app = CreateObject("LeCroy.XStreamDSO")

  ' Create the filename for the next panel setup to be recalled.
  app.SaveRecall.Setup.PanelFilename = "Setup89"

  ' Recall the panel setup from the named file.
  app.SaveRecall.Setup.DoRecallPanel
```

### DoSavePanel

### *Action*

#### Description

Saves the current panel settings to the previously specified file. If the filename already exists, the file will be overwritten without a prompt.

#### Example

```
' Visual Basic Script
  Set app = CreateObject("LeCroy.XStreamDSO")

  ' Create the filename for the next panel setup save.
  app.SaveRecall.Setup.PanelFilename = "TestSave"
```

```
' Save the panel setup to the named file.
app.SaveRecall.Setup.DoSavePanel
```

**InternalName1****String**

Range: Any number of characters

Description

Sets/Queries the name of internal panel setup memory 1.

**InternalName2****String**

Range: Any number of characters

Description

Sets/Queries the name of internal panel setup memory 2.

**InternalName3****String**

Range: Any number of characters

Description

Sets/Queries the name of internal panel setup memory 3.

**InternalName4****String**

Range: Any number of characters

Description

Sets/Queries the name of internal panel setup memory 4.

**InternalName5****String**

Range: Any number of characters

Description

Sets/Queries the name of internal panel setup memory 5.

**InternalName6****String**

Range: Any number of characters

Description

Sets/Queries the name of internal panel setup memory 6.

**PanelFilename****FileName**

Range: Any number of characters

Description

Sets/Queries the current filename for saving a panel setup. An ".lss" extension is automatically appended if not supplied.

**RecallInternal1****Action**

Description

Recalls the settings that are stored in internal panel memory 1.

## PART TWO: REFERENCE

---

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Recall the settings from internal panel memory 1.
app.SaveRecall.Setup.RecallInternal1
```

### RecallInternal2

**Action**

#### Description

Recalls the settings that are stored in internal panel memory 2.

### RecallInternal3

**Action**

#### Description

Recalls the settings that are stored in internal panel memory 3.

### RecallInternal4

**Action**

#### Description

Recalls the settings that are stored in internal panel memory 4.

### RecallInternal5

**Action**

#### Description

Recalls the settings that are stored in internal panel memory 5.

### RecallInternal6

**Action**

#### Description

Recalls the settings that are stored in internal panel memory 6.

### SaveInternal1

**Action**

#### Description

Saves the current instrument settings into internal panel memory 1.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Save the current settings into internal panel memory 1.
app.SaveRecall.Setup.SaveInternal1
```

### SaveInternal2

**Action**

#### Description

Saves the current instrument settings into internal panel memory 2.

**CHAPTER TWO****Control Reference**

---

**SaveInternal3****Action**

Description

Saves the current instrument settings into internal panel memory 3.

**SaveInternal4****Action**

Description

Saves the current instrument settings into internal panel memory 4.

**SaveInternal5****Action**

Description

Saves the current instrument settings into internal panel memory 5.

**SaveInternal6****Action**

Description

Saves the current instrument settings into internal panel memory 6.

## PART TWO: REFERENCE

## UTILITIES

*app.SaveRecall.Utilities*

These controls are used to manage files and folders, including the ability to create and delete folders, and to delete files.

**Utilities**

<b>CreateDir</b>	Action
<b>DeleteAll</b>	Action
<b>DeleteAllWithPrompt</b>	Action
<b>DeleteFile</b>	Action

**CreateDir****Action**

## Description

Creates the directory specified in the Directory control.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Create a named directory
app.SaveRecall.Utilities.Directory = "C:\MyDir"
app.SaveRecall.Utilities.CreateDir
```

**DeleteAll****Action**

## Description

Deletes all files in the directory specified by the Directory control without a cautionary prompt. Use with care! Files cannot be recovered if deleted accidentally.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Delete all files without showing a yes/no prompt.
app.SaveRecall.Utilities.Directory = "C:\MyDir"
app.SaveRecall.Utilities.DeleteAll
```

**DeleteAllWithPrompt****Action**

## Description

Deletes all files, but only after showing a prompt that allows the action to be abandoned. Until this prompt is acknowledged by the user, automation control is blocked.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Delete all files after showing a yes/no prompt.
app.SaveRecall.Utilities.Directory = "C:\MyDir"
app.SaveRecall.Utilities.DeleteAllWithPrompt
```

**DeleteFile****Action**

## Description

Deletes the file named by the Filename control.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Delete the named file
app.SaveRecall.Utilities.Filename = "C:\MyDir\MyFile.txt"
app.SaveRecall.Utilities.DeleteFile
```

## PART TWO: REFERENCE

**WAVEFORM***app.SaveRecall.Waveform*

Contains controls used for saving and recalling waveforms.

<b>Delimiter</b>	Enum
<b>DoRecall</b>	Action
<b>DoSave</b>	Action
<b>RecallDestination</b>	Enum
<b>RecallFrom</b>	Enum
<b>RecallSource</b>	Enum
<b>SaveDestination</b>	Enum
<b>SaveSource</b>	Enum
<b>SaveTo</b>	Enum
<b>SubFormat</b>	Enum
<b>TraceTitle</b>	String
<b>WaveFormat</b>	Enum
<b>WaveformDir</b>	FileName

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Save C1 into M1
app.SaveRecall.Waveform.SaveTo = "Memory"
app.SaveRecall.Waveform.SaveSource = "C1"
app.SaveRecall.Waveform.SaveDestination = "M1"
app.SaveRecall.Waveform.DoSave

' Save C2 into C:\MyDir in binary format
app.SaveRecall.Waveform.SaveTo = "File"
app.SaveRecall.Waveform.SaveSource = "C2"
app.SaveRecall.Waveform.WaveformDir = "C:\MyDir"
app.SaveRecall.Waveform.WaveFormat = "Binary"
app.SaveRecall.Waveform.DoSave

' Save C3 into C:\MyDir in text format
app.SaveRecall.Waveform.SaveTo = "File"
app.SaveRecall.Waveform.SaveSource = "C3"
app.SaveRecall.Waveform.WaveformDir = "C:\MyDir"
app.SaveRecall.Waveform.WaveFormat = "ASCII"
app.SaveRecall.Waveform.DoSave
```

**CHAPTER TWO****Control Reference****Delimiter****Enum**

## Description

Sets/Queries the delimiter to use when saving data in ASCII text mode.

## Values

Comma  
Semicolon  
Space  
Tab

**DoRecall****Action**

## Description

Recall waveform data into a trace memory. Source can be either an internal memory (M1 to M4), or a file on a mass-storage device, depending on the state of the "RecallFrom" control.

**DoSave****Action**

## Description

Save waveform data into an internal memory, or file on a mass-storage device, using the pre-specified source and destination.

**RecallDestination****Enum**

## Description

Sets/Queries the destination for waveform recall. When the DoRecall action is executed the waveform will be transferred into this destination trace.

## Values

M1  
M2  
M3  
M4

**RecallFrom****Enum**

## Description

Sets/Queries the type of source for waveform recall.

## Values

<b>File</b>	Recall from file on a mass-storage device
<b>Memory</b>	Recall from one of the internal memories (M1..M4)

**RecallSource****Enum**

## Description

Sets/Queries the source for recalling waveform data. Used only when recalling from an internal memory, with RecallSource = "Memory".

## Values

M1  
M2

## PART TWO: REFERENCE

---

M3

M4

### SaveDestination

*Enum*

#### Description

Sets/Queries the destination to which waveform data will be saved. Used only when SaveTo = "Memory".

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Setup to store trace C2 into M4 and perform the save operation

app.SaveRecall.Waveform.SaveTo = "Memory"
app.SaveRecall.Waveform.SaveSource = "C2"
app.SaveRecall.Waveform.SaveDestination = "M4"
```

#### Values

M1

M2

M3

M4

### SaveSource

*Enum*

#### Description

Sets/Queries the source from which waveform data will be saved.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the destination to memory for waveform save.
app.SaveRecall.Waveform.SaveTo = "Memory"
' Set the source to C2, for saving a waveform.
app.SaveRecall.Waveform.SaveSource = "C2"
' Set the destination to memory M4, for saving a waveform.
app.SaveRecall.Waveform.SaveDestination = "M4"
' Save waveform data as previously specified.
app.SaveRecall.Waveform.DoSave
```

#### Values

AllDisplayed

BadBits

Bits

C1

## CHAPTER TWO

## Control Reference

C2  
 C3  
 C4  
 Eye  
 F1  
 F2  
 F3  
 F4  
 F5  
 F6  
 F7  
 F8  
 M1  
 M2  
 M3  
 M4  
 PRBS

**SaveTo****Enum**

## Description

Sets/Queries the type of destination for waveform save.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the destination to Memory for waveform save.
app.SaveRecall.Waveform.SaveTo = "Memory"
```

## Values

<b>File</b>	Save into file on a mass-storage device
<b>Memory</b>	Save into an internal memory (M1..M4)

**SubFormat****Enum**

## Description

Sets/Queries the sub-format of the waveform data when saving in ASCII mode. Defines whether data is saved as Amplitude values only, Time and Amplitude pairs, or a standard format that includes the header.

## Values

AmplitudeOnly  
 Header  
 TimeAmplitude

**TraceTitle****String**

Range: Any number of characters.

## Description

Sets/Queries the title (prefix) to use when naming saved traces. This prefix will have the family (sequence) number appended to it when forming the filename.

**PART TWO: REFERENCE**

---

**WaveFormat****Enum**

## Description

Sets/Queries the format to use when saving waveform data into a file. "Binary" is the most efficient, storing one or two bytes per data sample, depending upon the number of significant bits. In ASCII mode, the Subformat and Delimiter controls define the data format.

## Values

<b>ASCII</b>	Plain ASCII files with choice of various delimiters
<b>Binary</b>	LeCroy's standard binary waveform format
<b>Excel</b>	
<b>MathCad</b>	
<b>MATLAB</b>	

**WaveformDir****FileName**

Range: Any number of characters.

## Description

Sets/Queries the directory for storing waveform files.

**SDA***app.SDA*

These controls are used for SDA (Serial Data Analyzer) models only.

<b>BERParamsOn</b>	Bool
<b>CalcType</b>	Enum
<b>ClockSource</b>	Enum
<b>DarkCalLevel</b>	Double
<b>DataSource</b>	Enum
<b>ErrorMapOn</b>	Bool
<b>EyeMode</b>	Enum
<b>FindFrequency</b>	Action
<b>FindScale</b>	Action
<b>HiPassFreq</b>	Double
<b>LowPassFreq</b>	Double
<b>MaskFailTraceOn</b>	Bool
<b>MaskFileName</b>	String
<b>NumPatternBits</b>	Integer
<b>PLLOn</b>	Bool
<b>RefReceiver</b>	Bool
<b>SDAMode</b>	Enum
<b>ShowFailLocation</b>	Bool
<b>SignalFrequency</b>	Double
<b>SignalMode</b>	Enum
<b>SignalType</b>	Enum
<b>StartN</b>	Integer
<b>StepN</b>	Integer
<b>StopN</b>	Integer
<b>Units</b>	Enum
<b>UserSignal</b>	Enum
<b>VerAutoFit</b>	Bool

**BERParamsOn****Bool**

Description

Sets/Queries whether the bit error rate parameters are shown.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the BER parameters on.
app.SDA.BERParamsOn = True
```

**CalcType****Enum**

Description

Sets/Queries the type of SDA calculation.

Example

## PART TWO: REFERENCE

---

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Sets the calculation type as peak to peak.
app.SDA.CalcType = "PeaktoPeak"
```

### Values

Average  
PeaktoPeak

### ClockSource

**Enum**

#### Description

Sets/Queries the clock source trace for SDA.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the instrument into SDA mode.
app.SDA.SDAMode = "MaskTest"

' Set the clock source as trace C2.
app.SDA.ClockSource = "C2"
```

### Values

C1  
C2  
C3  
C4  
F1  
F2  
F3  
F4  
F5  
F6  
F7  
F8  
M1  
M2  
M3  
M4  
PRBS

### DarkCalLevel

**Double**

Range: From -0.05 to 0.05, step 1e-006

#### Description

Sets/Queries the dark calibration level.

## CHAPTER TWO

## Control Reference

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the dark calibration level to 0.01.
app.SDA.DarkCalLevel = 0.01
' Visual Basic Script
```

## Description

Sets/Queries the data source trace for SDA.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the instrument into SDA mode.
app.SDA.SDAMode = "MaskTest"
' Set the data source as trace C3.
app.SDA.DataSource = "C3"
```

## Values

C1  
C2  
C3  
C4  
F1  
F2  
F3  
F4  
F5  
F6  
F7  
F8  
M1  
M2  
M3  
M4  
PRBS

**ErrorMapOn****Bool**

## Description

Sets/Queries whether the error map is on.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

**PART TWO: REFERENCE**

---

```
' Set the error map on.
app.SDA.ErrorMapOn = True
```

**EyeMode****Enum**

## Description

Sets/Queries the type of eye diagram.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the eye mode as traditional.
app.SDA.EyeMode = "Traditional"
```

## Values

Sequential  
Traditional

**FindFrequency****Action**

## Description

Instruct the instrument to find the frequency of the signal.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Find the frequency.
app.SDA.FindFrequency
```

**FindScale****Action**

## Description

Find the scale for SDA Jitter NCycle vs. N Plot.

**HiPassFreq****Double**

Range: From 10 to 1e+010, step 1

## Description

Set the SDA jitter filter high-pass frequency.

**LowPassFreq****Double**

Range: From 10 to 1e+010, step 1

## Description

Set the SDA jitter filter low-pass frequency.

**MaskFailTraceOn****Bool**

## Description

Sets/Queries whether the failed section of the trace should be shown.

## CHAPTER TWO

## Control Reference

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set mask fail trace on.
app.SDA.MaskFailTraceOn = True
```

**MaskFileName****String**

Range: Any number of characters.

## Description

Sets/Queries the current mask filename.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the instrument into SDA mode.
app.SDA.SDAMode = "MaskTest"

' Set the mask filename.
app.SDA.MaskFileName = "D:\XStreamMasks\FIB4181"
```

**NumPatternBits****Integer**

Range: From 1 to 2147483647, step 1

## Description

Number of bits in the data pattern. Used for the periodic jitter parameter.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the number of bits as 32.
app.SDA.NumPatternBits = 32
```

**PLLOn****Bool**

## Description

Sets/Queries the use of a PLL to track the clock frequency.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the PLL off.
app.SDA.PLLOn = False
```

**PART TWO: REFERENCE****RefReceiver****Bool**

## Description

Sets/Queries the status of a reference receiver. With an optical-to-electrical converter connected to the instrument, this variable enables or disables the use of information from the receiver.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set reference receiver as on.
app.SDA.RefReceiver = True
```

**SDAMode****Enum**

## Description

Sets/Queries the mode of operation of the instrument. Values: Scope, MaskTest, Jitter, BER, Clock, Summary for SDA+ASDA. Values: Scope, MaskTest for SDM

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the instrument into Mask Test mode.
app.SDA.SDAMode = "MaskTest"
```

## Values (SDA, ASDA)

- Scope
- MaskTest
- Jitter
- BER
- Clock
- Summary

## Values (SDM)

- Scope
- MaskTest

**ShowFailLocation****Bool**

## Description

Sets/Queries whether mask failures are to be shown by markers.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Show locations of failures.
app.SDA.ShowFailLocation = True
```

**CHAPTER TWO****Control Reference****SignalFrequency***Double*

Range: From 5e+007 to 4e+010, step 1

## Description

Sets/Queries the signal frequencies.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the signal frequency to 15 MHz.
app.SDA.SignalFrequency = 15e6
```

**SignalMode***Enum*

## Description

Values for this control depend upon the SDA standard set by the "SignalType" control.

## Values

Receiver  
TransAbs  
TransNrm

**SignalType***Enum*

## Description

Sets/Queries the signal type for SDA.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the instrument into SDA mode.
app.SDA.SDAMode = "MaskTest"

' Set signal type as STM16.
app.SDA.SignalType = "STM16"
```

## Values

1000BaseCX  
1000BaseLX  
1000BaseSX  
1000baseX  
10GBASELX4  
Custom  
DVI  
FC1063  
FC133  
FC2125  
FC266  
FC531  
IEEE1394b

**PART TWO: REFERENCE**

---

Infini2.5Gbs  
 OC1  
 OC12  
 OC3  
 OC48  
 PCIExpress  
 SATA1.5  
 STM1  
 STM16  
 STM4  
 STS1Eye  
 STS3Interface  
 STS3Transmit  
 USB2.0  
 XAUI

**StartN***Integer*

Range: From 1 to 100000, step 1

Description

Sets/Queries the unit interval where measurement is to start.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the UI where measurement is to start.
app.SDA.StartN =5
```

**StepN***Integer*

Range: From 1 to 100000, step 1

Description

Sets/Queries the gap between measurement groups, in unit intervals.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the step between measurements.
app.SDA.StepN = 2
```

**StopN***Integer*

Range: From 1 to 100000, step 1

Description

Sets/Queries the unit interval where measurement is to stop.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Set the UI where measurement is to stop.
app.SDA.StopN = 5000
```

**Units****Enum**

## Description

Sets/Queries the unit of measure.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the units as UI
app.SDA.Units = "UI"
```

## Values

S  
UI

**UserSignal****Enum**

## Description

A trace selected by the user to be displayed with horizontal scaling set by the Mask Test failed bit locator. For example, data source is C1 and user signal is C2. There is a mask failure on the data signal C2 between 40 ns and 80 ns. C2 would zoom to 40 ns–80 ns.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the instrument into SDA mode.
app.SDA.SDAMode = "MaskTest"

' Set the user signal source as trace C4.
app.SDA.UserSignal = "C4"
```

## Values

C1  
C2  
C3  
C4  
F1  
F2  
F3  
F4  
F5  
F6  
F7  
F8

## PART TWO: REFERENCE

---

M1  
M2  
M3  
M4  
NONE

### **VerAutoFit**

***Bool***

#### Description

Controls whether data is fit vertically to the mask automatically.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set vertical auto fitting off.
app.SDA.VerAutoFit = False
```

**BADBITS***app.SDA.BadBits*

<b>AbsLevel</b>	Double
<b>AxisXRotation</b>	Integer
<b>AxisYRotation</b>	Integer
<b>BitsInLocator</b>	Integer
<b>C1ReceiverStandard</b>	Enum
<b>C2ReceiverStandard</b>	Enum
<b>C3ReceiverStandard</b>	Enum
<b>C4ReceiverStandard</b>	Enum
<b>ClearSweeps</b>	Action
<b>ClockTIESlope</b>	Enum
<b>DarkCal</b>	Action
<b>DarkCalLevel</b>	Double
<b>DataSource</b>	Enum
<b>EyeMode</b>	Enum
<b>EyeThresholdType</b>	Enum
<b>FailCursorsOn</b>	Bool
<b>FailedList</b>	Enum
<b>FailedSymbolsFilter</b>	Enum
<b>LabelsPosition</b>	String
<b>LabelsText</b>	String
<b>MaskFailTraceOn</b>	Bool
<b>MaskFailX</b>	Double
<b>MaskFailY</b>	Double
<b>MaskType</b>	Enum
<b>MaxFailures</b>	Integer
<b>MeasurementMode</b>	Enum
<b>MonochromeEye</b>	Enum
<b>PercentLevel</b>	Integer
<b>Persist3DQuality</b>	Enum
<b>Persisted</b>	Bool
<b>Persistence3d</b>	Bool
<b>PersistenceMonoChrome</b>	Bool
<b>PersistenceSaturation</b>	Integer
<b>PersistenceTime</b>	Enum
<b>RefReceiver</b>	Bool
<b>SDAMode</b>	Enum
<b>ShowFailLocation</b>	Bool
<b>ShowLastTrace</b>	Bool
<b>SignalFrequency</b>	Double
<b>SignalType</b>	Enum
<b>SliceWidth</b>	Integer
<b>Stop</b>	Bool
<b>TrackMaskFail</b>	Action
<b>UseDotJoin</b>	Bool
<b>UseGrid</b>	String
<b>UserSignal</b>	Enum
<b>VerAutoFit</b>	Bool
<b>View</b>	Bool
<b>ViewLabels</b>	Bool
<b>XMargin</b>	Integer
<b>YMargin</b>	Integer

## PART TWO: REFERENCE

---

### AbsLevel

*Double*

Range: From -100 to 100, step 1e-007

Description

Voltage level to set in the TIE at level.

### AxisXRotation

*Integer*

Range: From -90 to 90, step 1

Description

Refer to the corresponding variable in **Display**, using SDA.BadBits.Persisted.

### AxisYRotation

*Integer*

Range: From -90 to 90, step 1

Description

Refer to the corresponding variable in **Display**, using SDA.BadBits.Persisted.

### BitsInLocator

*Integer*

Range: From 1 to 101, step 1

Description

Number of bits to display in the Mask Test bottom grid, where the bit that failed the mask is displayed in the center of the grid. For example, BitsInLocator = 5 means that 2 bits before and 2 bits after the failing bit are going to be displayed in the bottom grid.

### C1ReceiverStandard

*Enum*

Description

Read only. Indicates what reference receiver filter the optical-to-electrical converter uses on the SDA signal.

Values

DISABLED  
FC1063  
FC2125  
L1000BASE  
OC12  
OC3  
OC48  
OTHER

### C2ReceiverStandard

*Enum*

Description

Read only. Indicates what reference receiver filter the optical-to-electrical converter uses on the SDA signal.

Values

DISABLED  
FC1063  
FC2125  
L1000BASE  
OC12

## CHAPTER TWO

## Control Reference

OC3  
OC48  
OTHER

**C3ReceiverStandard****Enum**

## Description

Read only. Indicates what reference receiver filter the optical-to-electrical converter uses on the SDA signal.

## Values

DISABLED  
FC1063  
FC2125  
L1000BASE  
OC12  
OC3  
OC48  
OTHER

**C4ReceiverStandard****Enum**

## Description

Read only. Indicates what reference receiver filter the optical-to-electrical converter uses on the SDA signal.

## Values

DISABLED  
FC1063  
FC2125  
L1000BASE  
OC12  
OC3  
OC48  
OTHER

**ClearSweeps****Action**

## Description

Refer to the corresponding variable in **Acquisition.Cx**, using SDA.BadBits.Persisted.

**ClockTIESlope****Enum**

## Description

Sets the preferred edge for eye diagram alignment.

## Values

Both  
Negative  
Positive

## PART TWO: REFERENCE

---

### DarkCal

**Action**

#### Description

Press to start a no-signal calibration of an optical-to-electrical probe.

### DarkCalLevel

**Double**

Range: From -0.05 to 0.05, step 1e-006

#### Description

A read only variable that shows the dark level after calibration.

### DataSource

**Enum**

#### Description

Using SDA.BadBits, refer to the corresponding variable in SDA.

#### Values

C1  
C2  
C3  
C4  
F1  
F2  
F3  
F4  
F5  
F6  
F7  
F8  
M1  
M2  
M3  
M4  
PRBS

### EyeMode

**Enum**

#### Description

Refer to the corresponding variable in **SDA**, using SDA.BadBits.

#### Values

Sequential  
Traditional

### EyeThresholdType

**Enum**

#### Description

Threshold type for Eye Diagram TIE level. Change the Eye Diagram TIE settings to match the expected eye crossings, for example, SDA default for eye crossing at 50%. You can set it to EyeThresholdType = Percent and PercentLevel=32.

#### Values

Absloute  
Percent

## CHAPTER TWO

## Control Reference

**FailCursorsOn****Bool**

## Description

Turns on round cursors surrounding points in the eye diagram that penetrate the mask.

**FailedList****Enum**

## Description

Reads the indices of the bits that failed the mask test.

**FailedSymbolsFilter****Enum**

## Description

Sets what failed indices to get in the FailedList. Values: All, NearXY. For NearXY, see **MaskFailX** and **MaskFailY**.

**LabelsPosition****String**

Range: Any number of characters.

## Description

Refer to the corresponding variable in **Acquisition.Cx**, using SDA.BadBits.Persisted.

**LabelsText****String**

Range: Any number of characters.

## Description

Refer to the corresponding variable in **Acquisition.Cx**, using SDA.BadBits.Persisted.

**MaskFailTraceOn****Bool**

## Description

Refer to the corresponding variable in **SDA**, using SDA.BadBits.

**MaskFailX****Double**

Range: From 0 to 1, step 0.01

## Description

Shows the relative horizontal position of the selected mask failure cursors. For example, the user clicks on a mask failure cursor in the middle of the display. MaskFailX shows the selected horizontal position that would be 0.5.

**MaskFailY****Double**

Range: From 0 to 1, step 0.01

## Description

Shows the relative vertical position of the selected mask failure cursors. For example, the user clicks on a mask failure cursor in the middle of the display.

**MaskType****Enum**

## Description

Refer to the corresponding variable in **SDA**, using SDA.BadBits.

## Values

Absolute

## PART TWO: REFERENCE

---

Normalized

### MaxFailures

*Integer*

Range: From 1 to 10000, step 1

Description

Sets the number of failed bits to display in FailedList.

### MeasurementMode

*Enum*

Description

Display a set of parameters measuring various properties of the eye diagram.

Values

Amplitude  
Eye  
Off  
Timing

### MonochromeEye

*Enum*

Values

Colored  
Monochrome

### PercentLevel

*Integer*

Range: From 0 to 100, step 1

### Persist3DQuality

*Enum*

Description

Refer to the corresponding variable in **Acquisition.Cx**, using SDA.BadBits.Persisted.

Values

Shaded  
Solid  
WireFrame

### Persisted

*Bool*

Description

Refer to the corresponding variable in **Acquisition.Cx**, using SDA.BadBits.Persisted.

### Persistence3d

*Bool*

Description

Refer to the corresponding variable in **Acquisition.Cx**, using SDA.BadBits.Persisted.

### PersistenceMonoChrome

*Bool*

Description

Refer to the corresponding variable in **Acquisition.Cx**, using SDA.BadBits.Persisted.

### PersistenceSaturation

*Integer*

Range: From 1 to 100, step 1

## CHAPTER TWO

## Control Reference

## Description

Refer to the corresponding variable in **Acquisition.Cx**, using SDA.BadBits.Persisted.

**PersistenceTime****Enum**

## Description

Refer to the corresponding variable in **Acquisition.Cx**, using SDA.BadBits.Persisted.

## Values

0.5s  
10s  
1s  
20s  
2s  
5s  
Infinite

**RefReceiver****Bool**

## Description

Refer to the corresponding variable in **SDA**, using SDA.BadBits.

**SDAMode****Enum**

## Description

Refer to the corresponding variable in **SDA**, using SDA.BadBits.

## Values

MaskTest  
Scope

**ShowFailLocation****Bool**

## Description

Refer to the corresponding variable in **SDA**, using SDA.BadBits.

**ShowLastTrace****Bool**

## Description

Refer to the corresponding variable in **Acquisition.Cx**, using SDA.BadBits.Persisted.

**SignalFrequency****Double**

Range: From 5e+007 to 4e+010, step 1

## Description

Refer to the corresponding variable in **SDA**, using SDA.BadBits.

**SignalType****Enum**

## Description

Refer to the corresponding variable in **SDA**, using SDA.BadBits.

## PART TWO: REFERENCE

---

### Values

1000BaseCX  
 1000BaseLX  
 1000BaseSX  
 1000baseX  
 10GBASELX4  
 Custom  
 DVI  
 FC1063  
 FC133  
 FC2125  
 FC266  
 FC531  
 IEEE1394b  
 Infini2.5Gbs  
 OC1  
 OC12  
 OC3  
 OC48  
 PCIExpress  
 SATA1.5  
 STM1  
 STM16  
 STM4  
 STS1Eye  
 STS3Interface  
 STS3Transmit  
 USB2.0  
 XAUI

### SliceWidth

*Integer*

Range: From 0 to 100, step 1

#### Description

Eye Diagram measurement aid. Applies a vertical slice around the middle of the eye diagram for narrowing measurement areas.

### Stop

*Bool*

#### Description

Stop acquisition when the signal penetrates the mask.

### TrackMaskFail

*Action*

#### Description

Refer to the corresponding variable in **SDA**, using SDA.BadBits.

### UseDotJoin

*Bool*

#### Description

Refer to the corresponding variable in Acquisition.Cx, using SDA.BadBits.

**CHAPTER TWO****Control Reference****UseGrid****String**

Range: Any number of characters

Description

Using SDA.BadBits.Persisted, please refer to the corresponding variable in Acquisition.Cx

**UserSignal****Enum**

Description

Using SDA.BadBits, refer to the corresponding variable in SDA.

Values

C1  
 C2  
 C3  
 C4  
 F1  
 F2  
 F3  
 F4  
 F5  
 F6  
 F7  
 F8  
 M1  
 M2  
 M3  
 M4  
 NONE

**VerAutoFit****Bool**

Description

Using SDA.BadBits, refer to the corresponding variable in SDA.

**View****Bool**

Description

Using SDA.BadBits.Persisted, please refer to the corresponding variable in Acquisition.Cx

**ViewLabels****Bool**

Description

Using SDA.BadBits.Persisted, please refer to the corresponding variable in Acquisition.Cx

**XMargin****Integer**

Range: From 0 to 100, step 1

Description

Inflate the mask horizontally. Values: 0-100 in percent. 0 Means original mask; 100% means the mask stretches all the way over the horizontal span.

## PART TWO: REFERENCE

---

### **YMargin**

*Integer*

Range: From 0 to 100, step 1

#### Description

Inflate the mask vertically. Values: 0-100 in percent. 0 Means original mask; 100% means the mask stretches all the way over the vertical

## RESULT

*app.SDA.BadBits.Out.Result*

<b>DataArray</b>	Property
<b>ExtendedStatus</b>	Property
<b>FirstEventTime</b>	Property
<b>HorizontalFrameStart</b>	Property
<b>HorizontalFrameStop</b>	Property
<b>HorizontalOffset</b>	Property
<b>HorizontalPerStep</b>	Property
<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>HorizontalVarianceArray</b>	Property
<b>HorizontalVariances</b>	Property
<b>IndexOfFirstSampleInFrame</b>	Property
<b>LastEventTime</b>	Property
<b>NumFrameDimensions</b>	Property
<b>NumSamplesInFrame</b>	Property
<b>Samples</b>	Property
<b>Status</b>	Property
<b>StatusDescription</b>	Property
<b>Sweeps</b>	Property
<b>UpdateTime</b>	Property
<b>VerticalFrameStart</b>	Property
<b>VerticalFrameStop</b>	Property
<b>VerticalMaxPossible</b>	Property
<b>VerticalMinPossible</b>	Property
<b>VerticalOffset</b>	Property
<b>VerticalPerStep</b>	Property
<b>VerticalResolution</b>	Property
<b>VerticalUnits</b>	Property

**DataArray***Property*

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**ExtendedStatus***Property*

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**FirstEventTime***Property*

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalFrameStart***Property*

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**PART TWO: REFERENCE**

---

**HorizontalFrameStop*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalOffset*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalPerStep*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalResolution*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalUnits*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalVarianceArray*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalVariances*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**IndexOfFirstSampleInFrame*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**LastEventTime*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.

**NumFrameDimensions*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**NumSamplesInFrame*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**CHAPTER TWO****Control Reference****Samples*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**Status*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**StatusDescription*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**Sweeps*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**UpdateTime*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalFrameStart*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalFrameStop*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalMaxPossible*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalMinPossible*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalOffset*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalPerStep*****Property***

Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**PART TWO: REFERENCE**

---

**VerticalResolution*****Property***

## Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalUnits*****Property***

## Description

Using SDA.BadBits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**BITS***app.SDA.Bits*

<b>AxisXRotation</b>	Integer
<b>AxisYRotation</b>	Integer
<b>ClearSweeps</b>	Action
<b>EyeMode</b>	Enum
<b>LabelsPosition</b>	String
<b>LabelsText</b>	String
<b>MaskType</b>	Enum
<b>Persist3DQuality</b>	Enum
<b>Persisted</b>	Bool
<b>Persistence3d</b>	Bool
<b>PersistenceMonoChrome</b>	Bool
<b>PersistenceSaturation</b>	Integer
<b>PersistenceTime</b>	Enum
<b>ShowLastTrace</b>	Bool
<b>SignalFrequency</b>	Double
<b>SignalType</b>	Enum
<b>UseDotJoin</b>	Bool
<b>UseGrid</b>	String
<b>VerAutoFit</b>	Bool
<b>View</b>	Bool
<b>ViewLabels</b>	Bool

**AxisXRotation***Integer*

Range: From -90 to 90, step 1

Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

**AxisYRotation***Integer*

Range: From -90 to 90, step 1

Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

**ClearSweeps***Action*

Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

**EyeMode***Enum*

Description

Using SDA.Bits, please refer to the corresponding variable in SDA.

Values

Sequential  
Traditional**LabelsPosition***String*

Range: Any number of characters

## PART TWO: REFERENCE

---

### Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

### LabelsText

**String**

Range: Any number of characters

### Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

### MaskType

**Enum**

### Description

Using SDA.Bits, please refer to the corresponding variable in SDA.

### Values

Absolute  
Normalized

### Persist3DQuality

**Enum**

### Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

### Values

Shaded  
Solid  
WireFrame

### Persisted

**Bool**

### Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

### Persistence3d

**Bool**

### Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

### PersistenceMonoChrome

**Bool**

### Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

### PersistenceSaturation

**Integer**

Range: From 1 to 100, step 1

### Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

### PersistenceTime

**Enum**

### Description

Using SDA.Bits.View, please refer to the corresponding variable in  
Acquisition.Cx

## CHAPTER TWO

## Control Reference

## Values

0.5s  
 10s  
 1s  
 20s  
 2s  
 5s  
 Infinite

**ShowLastTrace****Bool**

## Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

**SignalFrequency****Double**

Range: From 5e+007 to 4e+010, step 1

## Description

Sets/Queries the signal frequency.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the signal frequency to 2.13 MHz
app.SDA.Bits.SignalFrequency = 2.13e6
```

**SignalType****Enum**

## Description

Using SDA.Bits, please refer to the corresponding variable in SDA.

## Values

1000BaseCX  
 1000BaseLX  
 1000BaseSX  
 1000baseX  
 10GBASELX4  
 Custom  
 DVI  
 FC1063  
 FC133  
 FC2125  
 FC266  
 FC531  
 IEEE1394b  
 Infini2.5Gbs  
 OC1  
 OC12  
 OC3  
 OC48  
 PCIExpress

## PART TWO: REFERENCE

---

SATA1.5  
STM1  
STM16  
STM4  
STS1Eye  
STS3Interface  
STS3Transmit  
USB2.0  
XAUI

### **UseDotJoin**

***Bool***

#### Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

### **UseGrid**

***String***

Range: Any number of characters

#### Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

### **VerAutoFit**

***Bool***

#### Description

Using SDA.Bits, please refer to the corresponding variable in SDA.

### **View**

***Bool***

#### Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

### **ViewLabels**

***Bool***

#### Description

Using SDA.Bits.View, please refer to the corresponding variable in Acquisition.Cx

## RESULT

*app.SDA.Bits.Out.Result*

Properties of the type `xxxx.Out.Result.yyyy` are those of the last completed acquisition. They are not affected if other CVars are changed after that acquisition was completed. This distinction between "Out.Result" properties and other CVars is most important when the trigger mode is Single or Stopped. You should treat "Out.Result" properties as read-only.

<b>DataArray</b>	Property
<b>ExtendedStatus</b>	Property
<b>FirstEventTime</b>	Property
<b>HorizontalFrameStart</b>	Property
<b>HorizontalFrameStop</b>	Property
<b>HorizontalOffset</b>	Property
<b>HorizontalPerStep</b>	Property
<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>HorizontalVarianceArray</b>	Property
<b>HorizontalVariances</b>	Property
<b>IndexOfFirstSampleInFrame</b>	Property
<b>LastEventTime</b>	Property
<b>NumFrameDimensions</b>	Property
<b>NumSamplesInFrame</b>	Property
<b>Samples</b>	Property
<b>Status</b>	Property
<b>StatusDescription</b>	Property
<b>Sweeps</b>	Property
<b>UpdateTime</b>	Property
<b>VerticalFrameStart</b>	Property
<b>VerticalFrameStop</b>	Property
<b>VerticalMaxPossible</b>	Property
<b>VerticalMinPossible</b>	Property
<b>VerticalOffset</b>	Property
<b>VerticalPerStep</b>	Property
<b>VerticalResolution</b>	Property
<b>VerticalUnits</b>	Property

**DataArray***Property*

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**ExtendedStatus***Property*

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**FirstEventTime***Property*

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**PART TWO: REFERENCE**

---

**HorizontalFrameStart*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalFrameStop*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalOffset*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalPerStep*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalResolution*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalUnits*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalVarianceArray*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalVariances*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**IndexOfFirstSampleInFrame*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**LastEventTime*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**NumFrameDimensions*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**CHAPTER TWO****Control Reference****NumSamplesInFrame*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**Samples*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**Status*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**StatusDescription*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**Sweeps*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**UpdateTime*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalFrameStart*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalFrameStop*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalMaxPossible*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalMinPossible*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalOffset*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**PART TWO: REFERENCE**

---

**VerticalPerStep*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalResolution*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalUnits*****Property***

Description

Using SDA.Bits.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**EYE***app.SDA.Eye*

<b>AxisXRotation</b>	Integer
<b>AxisYRotation</b>	Integer
<b>ClearSweeps</b>	Action
<b>DataSource</b>	Enum
<b>EyeMode</b>	Enum
<b>LabelsPosition</b>	String
<b>LabelsText</b>	String
<b>MaskType</b>	Enum
<b>Persist3DQuality</b>	Enum
<b>Persisted</b>	Bool
<b>Persistence3d</b>	Bool
<b>PersistenceMonoChrome</b>	Bool
<b>PersistenceSaturation</b>	Integer
<b>PersistenceTime</b>	Enum
<b>ShowLastTrace</b>	Bool
<b>SignalFrequency</b>	Double
<b>SignalType</b>	Enum
<b>UseDotJoin</b>	Bool
<b>UseGrid</b>	String
<b>UseMonochrome</b>	Bool
<b>VerAutoFit</b>	Bool
<b>View</b>	Bool
<b>ViewLabels</b>	Bool

**AxisXRotation***Integer*

Range: From -90 to 90, step 1

Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

**AxisYRotation***Integer*

Range: From -90 to 90, step 1

Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

**ClearSweeps***Action*

Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

**DataSource***Enum*

Description

Sets/Queries the source of the data for the eye diagram.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

## PART TWO: REFERENCE

---

```
' Set the source for the eye diagram to input channel C3.
app.SDA.eye.DataSource = "C3"
```

### Values

C1  
C2  
C3  
C4  
F1  
F2  
F3  
F4  
F5  
F6  
F7  
F8  
M1  
M2  
M3  
M4  
PRBS

### EyeMode

**Enum**

#### Description

Using SDA.Eye, please refer to the corresponding variable in SDA.

#### Values

Sequential  
Traditional

### LabelsPosition

**String**

Range: Any number of characters.

#### Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

### LabelsText

**String**

Range: Any number of characters

#### Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

### MaskType

**Enum**

#### Description

Using SDA.Eye, please refer to the corresponding variable in SDA.

#### Values

Absolute  
Normalized

**CHAPTER TWO****Control Reference****Persist3DQuality****Enum**

## Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

## Values

Shaded  
Solid  
WireFrame

**Persisted****Bool**

## Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

**Persistence3d****Bool**

## Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

**PersistenceMonoChrome****Bool**

## Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

**PersistenceSaturation****Integer**

Range: From 1 to 100, step 1

## Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

**PersistenceTime****Enum**

## Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

## Values

0.5s  
10s  
1s  
20s  
2s  
5s  
Infinite

**ShowLastTrace****Bool**

## Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

**SignalFrequency****Double**

Range: From 5e+007 to 4e+010, step 1

**PART TWO: REFERENCE**

---

**Description**

Please see the corresponding variable in app.Measure.Px.Operator (ParamEngine = "Dperiod@level").

**SignalType****Enum****Description**

Using SDA.Eye, please refer to the corresponding variable in SDA.

**Values**

1000BaseCX  
 1000BaseLX  
 1000BaseSX  
 1000baseX  
 10GBASELX4  
 Custom  
 DVI  
 FC1063  
 FC133  
 FC2125  
 FC266  
 FC531  
 IEEE1394b  
 Infini2.5Gbs  
 OC1  
 OC12  
 OC3  
 OC48  
 PCIExpress  
 SATA1.5  
 STM1  
 STM16  
 STM4  
 STS1Eye  
 STS3Interface  
 STS3Transmit  
 USB2.0  
 XAUI

**UseDotJoin****Bool****Description**

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

**UseGrid****String**

Range: Any number of characters

**Description**

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

**UseMonochrome****Bool****Description**

Sets/Queries whether the display is monochrome.

## CHAPTER TWO

Control Reference

---

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set monochrome off.
app.SDA.Eye.UseMonochrome = False
```

**VerAutoFit****Bool**

## Description

Using SDA.Eye, please refer to the corresponding variable in SDA.

**View****Bool**

## Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

**ViewLabels****Bool**

## Description

Using SDA.Eye, please refer to the corresponding variable in Acquisition.Cx.

## PART TWO: REFERENCE

---

### RESULT

*app.SDA.Eye.Out.Result*

Properties of the type `xxxx.Out.Result.yyyy` are those of the last completed acquisition. They are not affected if other CVars are changed after that acquisition was completed. This distinction between "Out.Result" properties and other CVars is most important when the trigger mode is Single or Stopped. You should treat "Out.Result" properties as read-only.

<b>Columns</b>	Property
<b>DataArray</b>	Property
<b>FirstEventTime</b>	Property
<b>HorizontalFrameStart</b>	Property
<b>HorizontalFrameStop</b>	Property
<b>HorizontalOffset</b>	Property
<b>HorizontalPerColumn</b>	Property
<b>HorizontalUnits</b>	Property
<b>LastEventTime</b>	Property
<b>MaxPopulationInRectangle</b>	Property
<b>NumFrameDimensions</b>	Property
<b>PopulationOfRectangle</b>	Property
<b>Rows</b>	Property
<b>Sweeps</b>	Property
<b>UpdateTime</b>	Property
<b>VerticalFrameStart</b>	Property
<b>VerticalFrameStop</b>	Property
<b>VerticalOffset</b>	Property
<b>VerticalPerRow</b>	Property
<b>VerticalUnits</b>	Property

### Columns

*Property*

#### Description

Queries the number of columns in the display.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the number of columns.
Columns = app.SDA.Eye.Out.Result.Columns
```

### DataArray

*Property*

#### Description

Using `SDA.Eye.Out.Result`, please refer to the corresponding variable in `Acquisition.Cx.Out.Result`

### FirstEventTime

*Property*

#### Description

Using `SDA.Eye.Out.Result`, please refer to the corresponding variable in `Acquisition.Cx.Out.Result`

**CHAPTER TWO****Control Reference**

---

**HorizontalFrameStart***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalFrameStop***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalOffset***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**HorizontalPerColumn***Property*

## Description

Queries the column spacing.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the horizontal spacing.
HSpace = app.SDA.Eye.Out.Result.HorizontalPerColumn
```

**HorizontalUnits***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**LastEventTime***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**MaxPopulationInRectangle***Property*

## Description

Queries the highest population in the array.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the total population.
MaxPop = app.SDA.Eye.Out.Result.MaxPopulationInRectangle
```

**PART TWO: REFERENCE**

---

**NumFrameDimensions***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**PopulationOfRectangle***Property*

## Description

Queries the total population of the array.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the total population.
TotalPop = app.SDA.Eye.Out.Result.PopulationOfRectangle
```

**Rows***Property*

## Description

Queries the number of rows in the display.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the number of rows.
Rows = app.SDA.Eye.Out.Result.Rows
```

**Sweeps***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**UpdateTime***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalFrameStart***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalFrameStop***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**CHAPTER TWO****Control Reference**

---

**VerticalOffset***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalPerRow***Property*

## Description

Queries the vertical spacing.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Inspect the vertical spacing.
VSpace = app.SDA.Eye.Out.Result.VerticalPerRow
```

**VerticalUnits***Property*

## Description

Using SDA.Eye.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

## PART TWO: REFERENCE

## TIE

*app.SDA.TIE*

<b>AbsLevel</b>	Double
<b>EyeThresholdType</b>	Enum
<b>FindFrequency</b>	Action
<b>PercentLevel</b>	Integer
<b>PLLOn</b>	Bool
<b>SignalFrequency</b>	Double
<b>Slope</b>	Enum
<b>View</b>	Bool

**AbsLevel***Double*

Range: From -100 to 100, step 1e-007

Description

Please see the corresponding variable in app.Measure.Px.Operator (ParamEngine = "TIE@level")

**EyeThresholdType***Enum*

Description

Sets/Queries whether the eye threshold is measured in absolute units or percentage.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the eye threshold to percent.
app.SDA.TIE.EyeThresholdType = "Absolute"
```

Values

Absolute  
Percent

**FindFrequency***Action*

Description

Determine the signal frequency from the waveform data.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Calculate the signal frequency from the trace data.
app.SDA.TIE.FindFrequency
```

**PercentLevel***Integer*

Range: From 0 to 100, step 1

Description

Please see the corresponding variable in app.Measure.Px.Operator (ParamEngine = "TIE@level")

**CHAPTER TWO****Control Reference**

---

**PLLOn****Bool**

## Description

Please see the corresponding variable in app.SDA.

**SignalFrequency****Double**

Range: From 1000 to 1e+011, step 1

## Description

Sets/Queries the signal frequency for TIE.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the signal frequency to 2.00 MHz
app.SDA.TIE.SignalFrequency = 2.0e6
```

**Slope****Enum**

## Description

Please see the corresponding variable in app.Measure.Px.Operator (ParamEngine = "TIE@level")

## Values

Both  
Neg  
Pos

**View****Bool**

## Description

Sets/Queries the visibility of the function.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Show the TIE function.
app.SDA.TIE.View = True
```

## PART TWO: REFERENCE

---

### RESULT

*app.SDA.TIE.Out.Result*

Properties of the type xxxx.Out.Result.yyyy are those of the last completed acquisition. They are not affected if other cvars are changed after that acquisition was completed. This distinction between "Out.Result" properties and other cvars is most important when the trigger mode is Single or Stopped. You should treat "Out.Result" properties as read-only.

### Result

<b>ExtendedStatus</b>	Property
<b>FirstEventTime</b>	Property
<b>HorizontalResolution</b>	Property
<b>HorizontalUnits</b>	Property
<b>LastEventTime</b>	Property
<b>NumFrameDimensions</b>	Property
<b>Status</b>	Property
<b>StatusDescription</b>	Property
<b>UpdateTime</b>	Property
<b>Value</b>	Property
<b>ValueArray</b>	Property
<b>VerticalResolution</b>	Property
<b>VerticalUnits</b>	Property

### ExtendedStatus

*Property*

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

### FirstEventTime

*Property*

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

### HorizontalResolution

*Property*

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

### HorizontalUnits

*Property*

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

### LastEventTime

*Property*

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

### NumFrameDimensions

*Property*

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**CHAPTER TWO****Control Reference**

---

**Status*****Property***

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**StatusDescription*****Property***

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**UpdateTime*****Property***

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**Value*****Property***

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**ValueArray*****Property***

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalResolution*****Property***

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

**VerticalUnits*****Property***

Description

Using SDA.TIE.Out.Result, please refer to the corresponding variable in Acquisition.Cx.Out.Result

## PART TWO: REFERENCE

**DATETIMESETUP***app.Utility.DateTimeSetup*

This set of variables controls the date and time setup. In addition to manual controls for hh/mm/ss and dd/mm/yy, there is the ability to set the time and date from an Internet clock, using the SNTP protocol.

<b>CurrentDateAndTime</b>	String
<b>Day</b>	Integer
<b>Hour</b>	Integer
<b>Minute</b>	Integer
<b>Month</b>	Integer
<b>Second</b>	Integer
<b>SetFromSNTP</b>	Action
<b>Validate</b>	Action
<b>Year</b>	Integer

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set time/date from the NIST Internet clock
app.Utility.DateTimeSetup.SetFromSNTP
```

**CurrentDateAndTime***String*

Range: Any number of characters.

## Description

Reads the current date and time from the real-time calendar and clock.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the current date and time from the real-time calendar and
clock.
app.Utility.DateTimeSetup.CurrentDateAndTime
```

**Day***Integer*

Range: From 1 to 31, step 1

## Description

Sets/Queries the day of the month setting of the real-time clock as a number. The value will not be accepted by the clock until `app.Utility.DateTimeSetup.Validate` is sent. All time/date controls are validated at the same time.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the day of the month as 21.
```

**CHAPTER TWO****Control Reference**

---

```
app.Utility.DateTimeSetup.Day = 21
app.Utility.DateTimeSetup.Validate
```

**Hour****Integer**

Range: From 0 to 23, step 1

**Description**

Sets/Queries the hours setting of the real-time clock as a number. The value will not be accepted by the clock until `app.Utility.DateTimeSetup.Validate` is sent. All time/date controls are validated at the same time.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the hour as 13.
app.Utility.DateTimeSetup.Hour = 13
app.Utility.DateTimeSetup.Validate
```

**Minute****Integer**

Range: From 0 to 59, step 1

**Description**

Sets/Queries the minutes setting of the real-time clock as a number. The value will not be accepted by the clock until `app.Utility.DateTimeSetup.Validate` is sent. All time/date controls are validated at the same time.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the minute as 34.
app.Utility.DateTimeSetup.Minute = 34
app.Utility.DateTimeSetup.Validate
```

**Month****Integer**

Range: From 1 to 12, step 1

**Description**

Sets/Queries the month setting of the real-time clock as a number. The value will not be accepted by the clock until `app.Utility.DateTimeSetup.Validate` is sent. All time/date controls are validated at the same time.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the month as August.
app.Utility.DateTimeSetup.Month = 8
```

## PART TWO: REFERENCE

---

app.Utility.DateTimeSetup.Validate

### Second

*Integer*

Range: From 0 to 59, step 1

#### Description

Sets/Queries the seconds setting of the real-time clock as a number. The value will not be accepted by the clock until app.Utility.DateTimeSetup.Validate is sent. All time/date controls are validated at the same time.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the seconds as 55.
app.Utility.DateTimeSetup.Second = 55
app.Utility.DateTimeSetup.Validate
```

### SetFromSNTP

*Action*

#### Description

Sets the real time clock from the simple network time protocol.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the real time clock from the simple network time protocol.
app.Utility.DateTimeSetup.SetFromSNTP
```

### Validate

*Action*

#### Description

Validates any new settings. This action = clicking "Validate Changes" on the Date/Time page.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the day, hour, and minute, and validate.
app.Utility.DateTimeSetup.Day = 3
app.Utility.DateTimeSetup.Hour = 5
app.Utility.DateTimeSetup.Minute = 8

app.Utility.DateTimeSetup.Validate
```

**CHAPTER TWO****Control Reference**

---

**Year***Integer*

Range: From 2000 to 2099, step 1

**Description**

Sets/Queries the year setting of the real-time clock as a number. The value will not be accepted by the clock until `app.Utility.DateTimeSetup.Validate` is sent. All time/date controls are validated at the same time.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the year as 2003.
app.Utility.DateTimeSetup.Year = 2003
app.Utility.DateTimeSetup.Validate
```

## PART TWO: REFERENCE

## OPTIONS

*app.Utility.Options*

The options subsystem contains controls to query the list of installed software and hardware options.

<b>InstalledHWOptions</b>	String
<b>InstalledSWOptions</b>	String
<b>ScopeID</b>	String

**InstalledHWOptions****String**

Range: Any number of characters.

Description

Shows a list of the installed hardware options.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the list of installed hardware options and present
' in a popup dialog
MsgBox app.Utility.Options.InstalledHWOptions
```

**InstalledSWOptions****String**

Range: Any number of characters.

Description

Shows list of installed software options.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the list of installed software options and display
' in a popup dialog
MsgBox app.Utility.Options.InstalledSWOptions
```

**ScopeID****String**

Range: Any number of characters.

Description

Queries the ID of the instrument. This ID should be specified when purchasing software options for your instrument.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Read the ID of the instrument.
MsgBox app.Utility.Options.ScopeID
```

**REMOTE***app.Utility.Remote*

These controls are related to the remote control section of the instrument. In this context Automation is not considered part of "Remote". Remote control currently includes control using ASCII remote commands from GPIB or TCP/IP.

<b>AllowControlFrom</b>	String
<b>Assistant</b>	Enum
<b>GpibAddress</b>	Integer
<b>Interface</b>	Enum
<b>RestrictControl</b>	Enum
<b>SetToErrorsOnlyAndClearAtStartup</b>	Bool

**AllowControlFrom***String*

Range: Any number of characters.

Description

Sets/Queries an IP address from which remote control is allowed, if remote control has been restricted to specified clients using the RestrictControl control. This string contains a comma-delimited list of addresses, which can be supplied either in IP or DNS form. For example: "126.2.2.34,dansWorkstation"

**Assistant***Enum*

Description

Sets/Queries the setting of the remote assistant.

Values

<b>EO</b>	Log errors only
<b>FD</b>	Log all remote commands/queries
<b>OFF</b>	Turn the assistant off

**GpibAddress***Integer*

Range: From 1 to 30, step 1

Description

Queries the current GPIB address for remote control.

**Interface***Enum*

Description

Sets/Queries the currently selected type of remote control interface.

Values

GPIB  
Off  
TCPIP

**RestrictControl***Enum*

Description

Sets/Queries whether remote control is restricted to certain hosts, where the host name is defined either by IP address or DNS name.

## PART TWO: REFERENCE

---

### Values

No  
Yes

### **SetToErrorsOnlyAndClearAtStartup**

***Bool***

### Description

Enables the resetting of the remote assistant to "Errors Only" mode whenever the instrument is reset. Also ensures that the remote assistant log is cleared upon startup. This control is set by default to lower the risk that the remote assistant will be set to "Full Dialog" mode and be forgotten, causing a decrease in remote control performance.

**WEBEDITOR***app.WebEditor*

This set of variables controls the web editor, which shows the paths for data flow in the instrument. This feature is not supported on all instruments. Currently it is supported on DDA and SDA models, and models with XMAP and/or XMATH software options.

**WebEditor**

<b>AddConnection</b> ([in] VARIANT destProcessor, [in] VARIANT destInputPin, [in] VARIANT sourceProcessor, [in]	Method
<b>AddPreview</b> ([in] VARIANT sourceProcessor, [in] VARIANT sourcePin, [in] BSTR previewName, [in] double	Method
<b>AddProcessor</b> ([in] VARIANT processorOrClassId, [in] BSTR requestedName, [in] double xPosition, [in]	Method
<b>ClearSweeps</b>	Action
<b>GetProcessor</b> ([in] VARIANT processor)	Method
<b>RemoveAll</b> ()	Method
<b>RemoveConnection</b> ([in] VARIANT destProcessor, [in] VARIANT destInputPin)	Method
<b>RemovePreview</b> ([in] VARIANT processor)	Method
<b>RemoveProcessor</b> ([in] VARIANT processor)	Method

## Example

```

' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Enter auto-trigger mode
app.Acquisition.TriggerMode = "Auto"

' Show the web editor and remove all processors from it
app.ActiveView = "WebEdit"
app.WebEditor.RemoveAll

' Create a Waveform Adder, name it "MyAvg", and place it at
x=200, y=30
app.WebEditor.AddProcessor "LeCroy.WaveformAdder", "MyAdder",
200, 30

' Connect the output of channel 1 "C1Exec", to the first input of

' the adder, and the output of channel 2 "C2Exec" to the second
app.WebEditor.AddConnection "MyAdder", 0, "C1Exec", 0
app.WebEditor.AddConnection "MyAdder", 1, "C2Exec", 0

' Add a preview window to show the averaged output
app.WebEditor.AddPreview "MyAdder", 0, "MyAvgPreview", 370, 40

```

## PART TWO: REFERENCE

---

```
' Place function F1 into WebEdit mode, and hook the adder output
to it
app.Math.F1.View = True
```

**AddConnection([in] VARIANT destProcessor, [in] VARIANT destInputPin, [in] VARIANT sourceProcessor, [in] VARIANT sourceOutputPin)** *Method*

Description

Adds a connection between two 'pins' of nodes placed within the Web Editor. Pins are described by the name of the node, and the zero-based index of the pin on that node.

```
app.Math.F1.MathMode = "WebEdit"
```

**AddPreview([in] VARIANT sourceProcessor, [in] VARIANT sourcePin, [in] BSTR previewName, [in] double xPosition, [in] double yPosition)** *Method*

Description

Adds a Preview to the specified pin of the specified node. The coordinates specify where the preview will appear on the Web, with 0,0 being the top-left corner.

**AddProcessor([in] VARIANT processorOrClassId, [in] BSTR requestedName, [in] double xPosition, [in] double yPosition)** *Method*

Description

Adds a named "processor" to the web. To determine the name of a processor, just place it on the web using the GUI and hover the mouse over the node. The "ProgID" of the node, in the format 'LeCroy.<procName>' will appear. When processors are added from automation, there is no distinction between Measure, Math, and Pass/Fail processors.

**ClearSweeps** *Action*

Description

Clears any accumulated data for nodes such as Average, Persistence, etc. that reside in the processing web.

**GetProcessor([in] VARIANT processor)** *Method*

Description

Retrieves a reference to a processor that has been added to the Web. This reference may then be used to access the processor's controls. See the Math/Measure control reference section of this manual for a list of the available controls for each processor.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Show the web editor and remove all processors from it
app.ActiveView = "WebEdit"
app.WebEditor.RemoveAll

' Create a Waveform Averager, name it "MyAvg", and place it at
```

```

x=200, y=30
app.WebEditor.AddProcessor "LeCroy.Average", "MyAvg", 200, 30

' Retrieve a pointer to the averager and set it's number of
sweeps
' to the value 1234
set myAverager = app.WebEditor.GetProcessor("MyAvg")

```

**RemoveAll()****Method**

## Description

Removes all processors from the web.

**RemoveConnection([in] VARIANT destProcessor, [in] VARIANT destInputPin)****Method**

## Description

Removes a connection between two pins on the web.

**RemovePreview([in] VARIANT processor)****Method**

## Description

Removes the named preview display.

**RemoveProcessor([in] VARIANT processor)****Method**

## Description

Removes the named processor from the Web.

**X-STREAM**

**PART TWO: REFERENCE**

---

BLANK PAGE

# **MATH/MEASURE CONTROL REFERENCE**

This section provides a reference guide for all controls used by the various 'processors' available in the instrument's Math and Measure subsystems.

## PART TWO: REFERENCE

### Math/Measure Control Reference

This section of the manual is concerned with documenting the controls used by the numerous Math/Measure “processors” in the X-Stream DSO. The basic structure is very similar to that used in the preceding section, with one basic exception.

In the Math and Measure sections of the DSO, the automation hierarchy is more dynamic than it is elsewhere. When a math or measure operator is selected, its controls “appear” in the hierarchy, switching out the controls used by the previously selected operator.

For example, when the FFT operator is selected as Operator 1 of Math Function F1:

```
app.Math.F1.Operator1 = "FFT"
```

The following controls will be available:

```
app.Math.F1.Operator1Setup.Algorithm
app.Math.F1.Operator1Setup.FillType
app.Math.F1.Operator1Setup.SuppressDC
app.Math.F1.Operator1Setup.Type
app.Math.F1.Operator1Setup.Window
```

The nodes in the automation hierarchy that behave in this way include:

```
app.Math.Fx.Operator1Setup
app.Math.Fx.Operator2Setup

app.Measure.Px.Operator

app.PassFail.Q1.Operator
```

In order to clarify this dependency, the titles of each of the sections in this chapter follow this format:

**AVERAGE**

Processor  
Name

***app.Math.Fx.OperatorYSetup (Operator = "Average")***

Path to controls, where:  
Fx := { F1, F2, F3, F4, F5, F6, F7, F8 } and  
OperatorYSetup := { Operator1Setup, Operator2Setup }

**AVERAGE***app.Math.Fx.OperatorYSetup (Operator = "Average")*

## Description

Waveform Averaging.

<b>AverageType</b>	Enum
<b>ClearSweeps</b>	Action
<b>Sweeps</b>	Integer

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Turn trace F1 on and setup to average the data from C1
' Average mode is set to Continuous
app.Math.F1.View = True
app.Math.F1.Operator1 = "Average"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Source1 = "C1"
app.Math.F1.Operator1Setup.AverageType = "Continuous"
app.Math.F1.Operator1Setup.Sweeps = 50
```

**AverageType****Enum**

## Description

Sets/Queries the averaging mode. Continuous and Summation modes are supported.

## Values

Continuous  
Summed

**ClearSweeps****Action**

## Description

Clears all averaged sweeps.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Clear sweeps for average in trace F1.
app.Math.F1.Operator1Setup.ClearSweeps
```

## PART TWO: REFERENCE

---

### Sweeps

*Integer*

Range: From 1 to 1000000, step 1

#### Description

Sets/Queries the number of sweeps to be averaged when trace Fx is set to averaging: continuous or summed.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set number of sweeps to be averaged in trace F1 as 20.
app.Math.F1.Operator1Setup.Sweeps = 20
```

**BOXCAR***app.Math.Fx.OperatorYSetup (Operator = "Boxcar")*

Rectangular BoxCar filter (local running average) of specified length.

**Length***Integer*

Range: From 2 to 50, step 1

**Description**

Sets / Queries the length, in samples, of the boxcar FIR filter (i.e., the running average of a local set of "length" points)

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the filter length for the boxcar function in trace F1
app.Math.F1.View = True
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Boxcar"
app.Math.F1.Operator1Setup.Length = 20
```

## PART TWO: REFERENCE

**CORRELATION** *app.Math.Fx.OperatorYSetup (Operator = "Correlation")*

Correlates a portion of one waveform to another.

<b>CorrLength</b>	Double
<b>CorrStart</b>	Double

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure correlation in F3 using a length of 3.5 divisions,
' starting at the first division.
app.Math.F3.View = True
app.Math.F3.Operator1 = "Correlation"
app.Math.F3.Operator1Setup.CorrLength = 3.5
app.Math.F3.Operator1Setup.CorrStart = 1
```

**CorrLength****Double**

Range: From 0.001 to 10, step 0.001

## Description

Sets/Queries the length (in graticule divisions) of the section of the first input trace that is used in the calculation of correlation.

**CorrStart****Double**

Range: From 0 to 9.999, step 0.001

## Description

Sets/Queries the position (in graticule divisions) of the start of the section of trace 1 that is used in the correlation function in trace Fx.

**DERIVATIVE***app.Math.Fx.OperatorYSetup (Operator = "Derivative")*

Computes the derivative of the waveform:  $(\text{next\_sample\_value} - \text{this\_sample\_value}) / \text{horizontal\_sample\_interval}$ .

<b>EnableAutoScale</b>	Bool
<b>FindScale</b>	Action
<b>VerOffset</b>	Double
<b>VerScale</b>	DoubleLockstep

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Start a find scale operation for derivative function trace F1
app.Math.F1.View = True
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Derivative"
app.Math.F1.Operator1Setup.FindScale
```

**EnableAutoScale****Bool**

## Description

Sets/Queries whether the Autoscale function is enabled for derivative function trace Fx. If enabled, an Autoscale operation is performed whenever the setup changes.

**FindScale****Action**

## Description

Initiates a Find Scale action, to set a suitable vertical scale for the derivative function trace Fx.

**VerOffset****Double**

Range: From -1e+006 to 1e+006, step 1e-009

## Description

Sets/Queries the vertical offset of derivative function trace Fx.

**VerScale****DoubleLock step**

Range: From 1e-012 to 1e+013, step 10000, locked to 1-2-5

## Description

Sets/Queries the vertical scale of derivative function Fx.

**PART TWO: REFERENCE**

---

**DESKEW***app.Math.Fx.OperatorYSetup (Operator = "Deskew")*

Deskew waveform by shifting it in time.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set the displacement of the trace F3 to 3.7e-9
app.Math.F3.View = True
app.Math.F3.MathMode = "OneOperator"
app.Math.F3.Operator1 = "Deskew"
app.Math.F3.Operator1Setup.WaveDeskew = 3.7e-9
```

**WaveDeskew****Double**

Range: From -0.1 to 0.1, step 1e-012

**Description**

Sets/Queries the displacement in time of trace Fx. A positive value delays the signal; a negative one makes it appear earlier.

**ENVELOPE***app.Math.Fx.OperatorYSetup (Operator = "Envelope")*

Envelope of minimum and maximum values for an ensemble of sweeps.

<b>ClearSweeps</b>	Action
<b>Sweeps</b>	Integer

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F3 to be an envelope of C1
app.Math.F3.View = True
app.Math.F3.Source1 = "C1"
app.Math.F3.MathMode = "OneOperator"
app.Math.F3.Operator1 = "Envelope"
app.Math.F3.Operator1Setup.Sweeps = 1000
```

**ClearSweeps****Action**

## Description

Initiates a Clear Sweeps operation for envelope function trace Fx.

**Sweeps****Integer**

Range: From 1 to 1000000, step 1

## Description

Sets/Queries the maximum number of sweeps to be used by envelope function trace Fx.

## PART TWO: REFERENCE

---

### **ERES**      *app.Math.Fx.OperatorYSetup (Operator = "EnhancedResolution")*

Enhanced resolution achieved through FIR filtering, using well behaved filters, with precalculated noise gain.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F3 to be an ERes of C1
app.Math.F3.View = True
app.Math.F3.Source1 = "C1"
app.Math.F3.MathMode = "OneOperator"
app.Math.F3.Operator1 = "Eres"
app.Math.F3.Operator1Setup.Bits = "2"
```

#### **Bits**

#### ***Enum***

#### Description

Sets/Queries the increase in resolution in bits for the enhanced resolution function. Be sure to specify the value as a string, and not as a floating point number, for example, "0.5", "1", "1.5", "2", "2.5", or "3".

#### Values

```
0.5
1
1.5
2
2.5
3
```

**EXCELMATH***app.Math.Fx.OperatorYSetup (Operator = "ExcelMath")*

Performs Math in Excel. Transfers 1 or 2 waveforms into Excel and reads the resulting waveform.

<b>AddChart</b>	Action
<b>AddLabels</b>	Action
<b>Advanced</b>	Bool
<b>ClearSheet</b>	Action
<b>CreateDemoSheet</b>	Action
<b>FindScale</b>	Action
<b>NewSheet</b>	Bool
<b>OutputCell</b>	String
<b>OutputEnable</b>	Bool
<b>OutputHeaderCell</b>	String
<b>Scaling</b>	Enum
<b>Source1Cell</b>	String
<b>Source1Enable</b>	Bool
<b>Source1HeaderCell</b>	String
<b>Source2Cell</b>	String
<b>Source2Enable</b>	Bool
<b>Source2HeaderCell</b>	String
<b>SpreadsheetFilename</b>	FileName
<b>Status</b>	String
<b>WithHeader</b>	Bool

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F3 to process C1 in Excel using a demo-sheet
app.Math.F3.View = True
app.Math.F3.Source1 = "C1"
app.Math.F3.MathMode = "OneOperator"
app.Math.F3.Operator1 = "ExcelMath"
app.Math.F3.Operator1Setup.CreateDemoSheet
```

**AddChart****Action**

## Description

Adds a chart to the current Excel spreadsheet.

**AddLabels****Action**

## Description

Adds labels to the cells of the array headers in the Excel spreadsheet.

## PART TWO: REFERENCE

---

### Advanced

**Bool**

#### Description

Enables/Disables/Queries the advanced Excel settings. By default, the cell ranges used to store the input waveform, and to retrieve the calculated waveform, are preset.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function trace F3 to be an Excel function.
app.Math.F3.Operator1 = "ExcelMath"

' Enable the use of the advanced settings.
app.Math.F3.Operator1Setup.Advanced = True
```

### ClearSheet

**Action**

#### Description

Clears the contents of the current Excel spreadsheet.

### CreateDemoSheet

**Action**

#### Description

Creates a "demo sheet," an excel spreadsheet pre-labeled, and with the output column equation preset to invert the input data.

### FindScale

**Action**

#### Description

Sets a suitable scale for the output data from Excel on the instrument graticule when scaling has been set to manual.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function trace F3 to be an Excel function.
app.Math.F3.Operator1 = "ExcelMath"

' Set the scaling from the Excel spreadsheet to automatic.
app.Math.F3.Operator1Setup.Scaling = "Manual"

' Find a suitable scale for the output data
' on the instrument graticule.
app.Math.F3.Operator1Setup.FindScale
```

**NewSheet****Bool**

## Description

Enables/Disables/Queries the creation of a new Excel spreadsheet. If a new sheet is not to be created, an existing file name must be specified in the SpreadsheetFilename control.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function trace F3 to be an Excel function.
app.Math.F3.Operator1 = "ExcelMath"

' Enable the creation of a new Excel spreadsheet.
app.Math.F3.Operator1Setup.NewSheet = True
```

**OutputCell****String**

Range: Any number of characters

## Description

Sets/Queries the cell label for output in the Excel function Fx. This cell marks the start (top) of the array of data to be taken from Excel into the instrument.

**OutputEnable****Bool**

## Description

Enables/Disables/Queries the transfer of output data from Excel to the instrument. If a one-way computation is required, where results of the Excel processing are not required, this should be set to False to increase performance.

**OutputHeaderCell****String**

Range: Any number of characters

## Description

Sets/Queries the header cell label for output in Excel function Fx. This is the starting cell for the header that carries setup information about the output waveform, from Excel to the instrument. Only used if the WithHeader control is set to True.

## PART TWO: REFERENCE

---

### Scaling

**Enum**

Description

Sets/Queries the method of scaling the output trace from the Excel spreadsheet.

Values

<b>Automatic</b>	Automatically scale the output waveform to full-scale
<b>FromSheet</b>	Retrieve scaling information from the output header in the spreadsheet
<b>Manual</b>	Manually auto-scale when FindScale is pressed

### Source1Cell

**String**

Range: Any number of characters

Description

Sets/Queries the cell label for source 1 in Excel function Fx. This cell marks the start (top) of the array into which data from the first source waveform is transferred.

### Source1Enable

**Bool**

Description

Enables/Disables/Queries the transfer of source 1 data from the instrument to Excel.

### Source1HeaderCell

**String**

Range: Any number of characters

Description

Sets/Queries the header cell label for source 1 in Excel function Fx. This is the starting cell for the header that carries setup information about waveform 1, from the instrument to Excel. This information includes waveform length, vertical and horizontal units, vertical and horizontal framing information, and horizontal scaling and offset information. Only used when the WithHeader control is set to True.

### Source2Cell

**String**

Range: Any number of characters

Description

Sets/Queries the cell label for source 2 in Excel function Fx. This cell marks the start (top) of the array into which data from the second source waveform is transferred.

### Source2Enable

**Bool**

Description

Enables/Disables/Queries the transfer of source 2 data from the instrument to Excel.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Set function trace F1 to be an Excel function.
app.Math.F1.Operator1 = "ExcelMath"

' Enable the transfer of source 2 data from the instrument to
Excel.
```

**Source2HeaderCell****String**

Range: Any number of characters

## Description

Sets/Queries the header cell label for source 2 in Excel function Fx. This is the starting cell for the header that carries setup information about waveform 2 from the instrument to Excel. This information includes waveform length, vertical and horizontal units, vertical and horizontal framing information, and horizontal scaling and offset information. Only used when the WithHeader control is set to True.

**SpreadsheetFilename****FileName**

Range: Any number of characters

## Description

Sets/Queries the file name of the current Excel spreadsheet.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function trace F3 to be an Excel function.
app.Math.F3.Operator1 = "ExcelMath"

' Disable the creation of a new Excel spreadsheet.
app.Math.F3.Operator1Setup.NewSheet = False

' Select the filename of the existing Excel spreadsheet to be
used.
```

**Status****String**

Range: Any number of characters

## Description

Inspects the status of the Excel-and-instrument combination. Examples are "OK", or "Excel not installed".

## PART TWO: REFERENCE

---

**WithHeader**

***Bool***

Description

Enables/Disables/Queries the presence of headers with the waveform

**FFT***app.Math.Fx.OperatorYSetup (Operator = "FFT")*

Fast Fourier Transform of waveform data.

<b>Algorithm</b>	Enum
<b>FillType</b>	Enum
<b>SuppressDC</b>	Bool
<b>Type</b>	Enum
<b>Window</b>	Enum

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F3 to perform an FFT of C1
app.Math.F3.View = True
app.Math.F3.Source1 = "C1"
app.Math.F3.MathMode = "OneOperator"
app.Math.F3.Operator1 = "FFT"
app.Math.F3.Operator1Setup.Algorithm = "Power2"
app.Math.F3.Operator1Setup.Window = "VonHann"
app.Math.F3.Operator1Setup.Type = "PowerSpectrum"
```

**Algorithm****Enum****Description**

Sets/Queries the algorithm for the FFT in function trace Fx.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function trace F4 to FFT.
app.Math.F4.Operator1 = "FFT"

' Set the FFT algorithm to power of two.
app.Math.F4.Operator1Setup.Algorithm = "Power2"
```

**Values**

```
LeastPrime
Power2
```

**FillType****Enum****Description**

Sets/Queries the type of trace fill to use in the FFT function trace Fx.

## PART TWO: REFERENCE

---

### Values

Truncate  
ZeroFill

### SuppressDC

**Bool**

#### Description

Enables/Disables suppression of the value at zero frequency in the FFT spectrum.

### Type

**Enum**

#### Description

Sets/Queries the type of FFT spectrum for function trace Fx.

### Values

<b>Imaginary</b>	Imaginary part of the complex spectrum
<b>Magnitude</b>	Magnitude with linear vertical scale
<b>Phase</b>	Phase
<b>PowerDensity</b>	Power Density
<b>PowerSpectrum</b>	Power Spectrum
<b>Real</b>	Real part of the complex spectrum

### Window

**Enum**

#### Description

Sets/Queries the type of window for FFT function trace Fx.

### Values

BlackmanHarris  
FlatTop  
Hamming  
Rectangular  
VonHann

**FILTER***app.Math.Fx.OperatorYSetup (Operator = "Filter")*

Processes waveform using specified digital filter.

<b>AutoLength</b>	Bool
<b>CosineBeta</b>	Double
<b>CustomFilename</b>	FileName
<b>FilterKind</b>	Enum
<b>FilterType</b>	Enum
<b>FirOrIir</b>	Enum
<b>GaussianBT</b>	Double
<b>HighFreqPass</b>	Double
<b>LowFreqPass</b>	Double
<b>NumberOfTaps</b>	Integer
<b>PassBandAttenuation</b>	Double
<b>PassBandRipple</b>	Double
<b>Rolloff</b>	Double
<b>StopBandAttenuation</b>	Double
<b>TransitionWidth</b>	Double
<b>Window</b>	Enum

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to filter C1
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Filter"
app.Math.F1.Operator1Setup.FirOrIir = "FIR"
app.Math.F1.Operator1Setup.FilterKind = "LowPass"
app.Math.F1.Operator1Setup.AutoLength = True
```

**AutoLength****Bool**

## Description

Enables/Disables/Queries status of the auto-length feature for the filter.

**CosineBeta****Double**

Range: From 0 to 100, step 1

## Description

Sets/Queries the constant beta for the raised root cosine filter Fx, as a percentage.

## PART TWO: REFERENCE

---

### CustomFilename

*FileName*

Range: Any number of characters

Description

Sets/Queries the name of the current custom file for filter Fx. Valid only when FilterKind is set to "Custom".

### FilterKind

*Enum*

Description

Sets/Queries kind of filter to use in function Fx.

Values

BandPass  
 BandStop  
 Custom  
 Gaussian  
 HighPass  
 LowPass  
 RaisedCosine  
 RaisedRootCosine

### FilterType

*Enum*

Description

Sets/Queries the type of filter to use in function Fx.

Values

Bessel  
 Butterworth  
 Chebyshev  
 InverseChebyshev

### FirOrIir

*Enum*

Description

Sets/Queries whether filter Fx is an FIR filter or an IIR filter.

Values

<b>FIR</b>	Finite Impulse Response Filter
<b>IIR</b>	Infinite Impulse Response Filter

**GaussianBT***Double*

Range: From 0 to 100, step 1

Description

Sets/Queries the value of the Gaussian BT constant for filter Fx. Valid only when FilterKind is set to "Gaussian".

**HighFreqPass***Double*

Range: From 1000 to 1e+010, step 1

Description

Sets/Queries the higher cut-off frequency for high-pass filter Fx.

**LowFreqPass***Double*

Range: From 1000 to 1e+010, step 1

Description

Sets/Queries the lower cut-off frequency for band-pass filter Fx.

**NumberOfTaps***Integer*

Range: From 0 to 2001, step 1

Description

Sets/Queries the number of taps in filter Fx. Valid only when the AutoLength control is set to False.

**PassBandAttenuation***Double*

Range: From 0.5 to 20, step 0.1

Description

Sets/Queries the pass-band attenuation of filter Fx.

**PassBandRipple***Double*

Range: From 0.5 to 20 step 0.1

Description

Sets/Queries the pass-band ripple.

**PART TWO: REFERENCE**

---

**Rolloff*****Double***

Range: From 1 to 1000, step 0.1

Description

Sets/Queries the roll-off of filter Fx.

**StopBandAttenuation*****Double***

Range: From 10 to 100, step 0.001

Description

Sets/Queries the stop-band attenuation of filter Fx.

**TransitionWidth*****Double***

Range: From 0 to 1e+010, step 1

Description

Sets/Queries the width of the transition in the frequency spectrum of filter Fx.

**Window*****Enum***

Description

Sets/Queries the type of window for filter Fx.

## Values

- Bartlett
- Blackman
- Hamming
- Hanning
- Kaiser
- Rectangular

**FLOOR***app.Math.Fx.OperatorYSetup (Operator = "Floor")*

Most negative or minimum values for an ensemble of sweeps.

<b>ClearSweeps</b>	Action
<b>Sweeps</b>	Integer

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to measure the Floor of the first 1000
' sweeps of C1
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Floor"
app.Math.F1.Operator1Setup.Sweeps = 1000
```

**ClearSweeps****Action**

## Description

Initiates a Clear Sweeps action for Floor function trace Fx.

**Sweeps****Integer**

Range: From 1 to 1000000, step 1

## Description

Sets/Queries the maximum number of sweeps for Floor function trace Fx.

## PART TWO: REFERENCE

**HISTOGRAM***app.Math.Fx.OperatorYSetup (Operator = "Histogram")*

Histogram of the values of a parameter. Or, if a waveform is used as the input, histogram the waveform sample amplitudes.

<b>AutoFindScale</b>	Bool
<b>Bins</b>	DoubleLockstep
<b>Center</b>	Double
<b>ClearSweeps</b>	Action
<b>FindScale</b>	Action
<b>HorScale</b>	DoubleLockstep
<b>Values</b>	Integer
<b>VerScaleType</b>	Enum

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to histogram the first 200000 sample
' values from source waveform C1 into 50 bins.
' Auto find-scale is enabled.
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Histogram"
app.Math.F1.Operator1Setup.AutoFindScale = True
app.Math.F1.Operator1Setup.Bins = 50
app.Math.F1.Operator1Setup.Values = 200000
```

**AutoFindScale*****Bool***

## Description

Enables/Disables automatic scale setting for histogram function Fx.

**Bins*****DoubleLock step***

Range: From 20 to 2000, step 1, locked to 1-2-5

## Description

Sets/Queries the number of bins in histogram function Fx.

**Center****Double**

Range: From -1e+010 to 1e+010, step 1e-012

Description

Sets/Queries the horizontal value at the center of the graticule of histogram function Fx.

**ClearSweeps****Action**

Description

Clears the contents of all the bins of histogram function Fx.

**FindScale****Action**

Description

Creates a suitable horizontal position and scale to include all the non-empty bins of histogram Fx.

**HorScale****DoubleLock step**

Range: From 1e-012 to 1e+012, step 0.01, locked to 1-2-5

Description

Sets/Queries the horizontal scale in units per division for histogram function Fx. Use the FindScale control to automatically determine the scale by looking at the non-zero populated bins.

**Values****Integer**

Range: From 20 to 2000000000, step 1

Description

Sets/Queries the maximum number of values from the source result to include in the histogram function Fx.

**VerScaleType****Enum**

Description

Sets/Queries the way that the vertical scale is calculated as histogram Fx grows.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function F1 as histogram.
app.Math.F1.Operator1 = "Histogram"

' Set the vertical scale type to linear with constant maximum.
app.Math.F1.Operator1Setup.VerScaleType = "LinConstMax"
```

Values

<b>LinConstMax</b>	Linear scale with constant maximum value
<b>Linear</b>	Linear scale

## PART TWO: REFERENCE

## INTEGRAL

*app.Math.Fx.OperatorYSetup (Operator = "Integral")*

Integral of the linearly rescaled (multiplier and adder) input.

<b>Adder</b>	Double
<b>FindScale</b>	Action
<b>Multiplier</b>	Double
<b>VerOffset</b>	Double
<b>VerScale</b>	DoubleLockstep

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to integrate C1
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Integral"
app.Math.F1.Operator1Setup.Multiplier = 2
app.Math.F1.Operator1Setup.Adder = 0.1
app.Math.F1.Operator1Setup.FindScale
```

**Adder****Double**

Range: From -1e-009 to 1e-009, step 1e-012

Description

Sets/Queries the additive A for integral function Fx.

**FindScale****Action**

Description

Initiates an action to find suitable vertical offset and scale for integral function trace Fx.

**Multiplier****Double**

Range: From -1e+006 to 1e+006, step 1e-006

Description

Sets/Queries the multiplying constant M for integral function Fx.

**VerOffset***Double*Range: From  $-1e+006$  to  $1e+006$ , step  $1e-012$ 

Description

Sets/Queries the vertical offset for integral function trace Fx.

**VerScale***DoubleLock step*Range: From  $1e-009$  to  $1e+007$  step  $0.01$ , locked to 1-2-5

Description

Sets/Queries the vertical scale for integral function trace Fx.

## PART TWO: REFERENCE

**INTERPOLATE** *app.Math.Fx.OperatorYSetup (Operator = "Interpolate")*

Interpolate, producing more points in the resulting waveform using linear, cubic, or weighted sin(x)/x algorithms.

<b>Expand</b>	DoubleLockstep
<b>InterpolateType</b>	Enum

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to interpolate C1
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Interpolate"
app.Math.F1.Operator1Setup.InterpolateType = "Cubic"
app.Math.F1.Operator1Setup.Expand = 5
```

**Expand*****DoubleLock step***

Range: From 2 to 50, step 0.1, locked to 1-2-5

**Description**

Sets/Queries the sampling expansion ratio for the interpolation function Fx.

**InterpolateType*****Enum*****Description**

Sets/Queries the type of interpolation for function trace Fx.

**Values**

Cubic  
Linear  
SinXX

**MATH SCRIPT** *app.Math.Fx.OperatorYSetup (Operator = "WaveScript")*

Visual basic script that produces a waveform from one or two input waveforms.

**Math script**

<b>Code</b>	String
<b>Language</b>	Enum
<b>Status</b>	String

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Place sample program code in a string
' This will invert the first 200 points of the waveform
code = "function Update()" + vbcrLf
code = code + "    unscaledData = InResult.DataArray(False)" +
vbcrLf
code = code + "    For i = 0 to 200" + vbcrLf
code = code + "        unscaledData(i) = -unscaledData(i)" +
vbcrLf
code = code + "    next" + vbcrLf
code = code + "    OutResult.DataArray(False) = unscaledData" +
vbcrLf
code = code + "end Function"

' Configure F1 to integrate C1
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Math Script"
```

**Code****String**

Range: Any number of characters

**Description**

Sets/Queries the scripting code used in math script function Fx. For complete details of programming the instrument in VBScript, please view the online Help and the relevant section of this manual.

**PART TWO: REFERENCE**

---

**Language*****Enum***

## Description

Sets/Queries the language used for math script function Fx.

## Values

<b>JScript</b>	Java Script
<b>VBScript</b>	Visual Basic Script

## Example

```
app.Math.F1.Operator1Setup.Language = "VBScript "  
app.Math.F1.Operator1Setup.Code = code
```

**Status*****String***

Range: Any number of characters

## Description

Inspects the status of the script operation. A typical message would be "Error at line 23, Type mismatch"; or "OK" if the supplied code executed successfully.

**MATHCADMATH** *app.Math.Fx.OperatorYSetup (Operator = "MathcadMath")*

Produces a waveform using a user specified Mathcad function.

<b>Advanced</b>	Bool
<b>FindScale</b>	Action
<b>NewSheet</b>	Bool
<b>OutputEnable</b>	Bool
<b>OutputHeaderVar</b>	String
<b>OutputVar</b>	String
<b>Reload</b>	Action
<b>Scaling</b>	Enum
<b>Source1Enable</b>	Bool
<b>Source1HeaderVar</b>	String
<b>Source1Var</b>	String
<b>Source2Enable</b>	Bool
<b>Source2HeaderVar</b>	String
<b>Source2Var</b>	String
<b>Status</b>	String
<b>WithHeader</b>	Bool
<b>WorksheetFilename</b>	FileName

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to process C1 using Mathcad
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "MathcadMath"
```

**Advanced****Bool**

## Description

Enables/Disables/Queries the use of the advanced features. When in advanced mode the names used for source and output vectors, in addition to names used for source and output headers, may be modified from their default values.

**FindScale****Action**

## Description

Sets a suitable vertical scale of the Mathcad output trace on the instrument graticule. Valid only when Manual scaling is specified.

## PART TWO: REFERENCE

---

### NewSheet

**Bool**

#### Description

Enables/Disables/Queries the creation of a new Mathcad worksheet.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function F1 as Mathcad calculation.
app.Math.F1.Operator1 = "MathcadMath"
' Enable creation of a new Mathcad worksheet.
app.Math.F1.Operator1Setup.NewSheet = True
```

### OutputEnable

**Bool**

#### Description

Enables/Disables/Queries the transmission of output data from Mathcad to the instrument.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function F1 as Mathcad calculation.
app.Math.F1.Operator1 = "MathcadMath"
' Enable transmission of output data.
app.Math.F1.Operator1Setup.OutputEnable = True
```

### OutputHeaderVar

**String**

Range: Any number of characters

#### Description

Sets/Queries the name in Mathcad of the output header variable.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function F1 as Mathcad calculation.
app.Math.F1.Operator1 = "MathcadMath"
' Enables use of headers.
app.Math.F1.Operator1Setup.WithHeader = True
' Sets the name of the output header variable
app.Math.F1.Operator1Setup.OutputHeaderVar = "header1"
```

**OutputVar****String**

Range: Any number of characters

## Description

Sets/Queries the name in Mathcad of the output variable.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function F1 as Mathcad calculation.
app.Math.F1.Operator1 = "MathcadMath"
' Sets the name of the output variable in Mathcad.
app.Math.F1.Operator1Setup.OutputVar = "output3"
```

**Reload****Action**

## Description

Reloads a specified Mathcad worksheet. If the worksheet does exist, the system creates an empty one with a name of the form "UntitledN", where N is an integer.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function F1 as Mathcad calculation.
app.Math.F1.Operator1 = "MathcadMath"
' Specifies a Mathcad worksheet name.
app.Math.F1.Operator1Setup.WorksheetFilename = "XStream34.mcd"
' Reload a Mathcad worksheet.
app.Math.F1.Operator1Setup.Reload
```

**Scaling****Enum**

## Description

Sets/Queries the method of vertical scaling of the Mathcad output trace on the instrument graticule.

## Values

Automatic  
Manual

**Source1Enable****Bool**

## Description

Enables/Disables/Queries the transmission of source 1 data from the instrument to Mathcad.

**PART TWO: REFERENCE**

---

<b>Source1HeaderVar</b>	<b>String</b>
Range: Any number of characters	
Description	
Sets/Queries the name in Mathcad of input 1 header variable.	
<b>Source1Var</b>	<b>String</b>
Range: Any number of characters	
Description	
Sets/Queries the name in Mathcad of input variable 1.	
<b>Source2Enable</b>	<b>Bool</b>
Description	
Enables/Disables/Queries the transmission of source 2 data from the instrument to Mathcad.	
<b>Source2HeaderVar</b>	<b>String</b>
Range: Any number of characters	
Description	
Sets/Queries the name in Mathcad of input 2 header variable.	
<b>Source2Var</b>	<b>String</b>
Range: Any number of characters	
Description	
Sets/Queries the name in Mathcad of input variable 2.	
<b>Status</b>	<b>String</b>
Range: Any number of characters	
Description	
Inspects the status of the Mathcad calculation.	
<b>WithHeader</b>	<b>Bool</b>
Description	
Enables/Disables/Queries inclusion of headers in the Mathcad calculation.	
<b>WorksheetFilename</b>	<b>FileName</b>
Range: Any number of characters	
Description	
Sets/Queries a Mathcad worksheet file name.	

**MATLAB MATH***app.Math.Fx.OperatorYSetup (Operator = "MATLABWaveform")*

Produces a waveform using a user specified MATLAB function.

<b>MATLABCode</b>	String
<b>MATLABPlot</b>	Bool
<b>MATLABScalePerDiv</b>	Double
<b>MATLABZeroOffset</b>	Double

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to process C1 using MATLAB
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "MATLAB math"
app.Math.F1.Operator1Setup.MATLABCode = "WformOut = -0.5 * WformIn"
```

**MATLABCode****String**

Range: Any number of characters

Description

Sends/Inspects the MATLAB code.

**MATLABPlot****Bool**

Description

Enables/Disables a MATLAB plot.

**MATLABScalePerDiv****Double**

Range: From 1e-009 to 1e+009, step 1e-009

Description

Sets/Queries the vertical scale in units per division, of the MATLAB output trace on the instrument graticule.

**PART TWO: REFERENCE**

---

**MATLABZeroOffset***Double*

Range: From -1e+009 to 1e+009, step 1e-009

## Description

Sets/Queries the vertical zero offset of the MATLAB output trace on the scope graticule. This is the position on the graticule where zero is found: a positive offset moves the trace downwards; a negative offset moves it upwards.

**PHISTOGRAM***app.Math.Fx.OperatorYSetup (Operator = "PersistenceHistogram")*

Histogram of a slice of a persistence map.

<b>CenterCursor</b>	Action
<b>ClearSweeps</b>	Action
<b>CutDirection</b>	Enum
<b>HorCutCenter</b>	Double
<b>HorCutWidth</b>	Double
<b>PctCutWidth</b>	Double
<b>VerCutCenter</b>	Double
<b>VerCutWidth</b>	Double

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to generate a slice of the persistence map of C1
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "PHistogram"
app.Math.F1.Operator1Setup.CutDirection = "Horizontal"
app.Math.F1.Operator1Setup.HorCutWidth = 10e-3
app.Math.F1.Operator1Setup.CenterCursor
```

**CenterCursor****Action**

## Description

Centralizes the cut on the graticule for persistence histogram Fx.

**ClearSweeps****Action**

## Description

Initiates a clearance of the contents of persistence histogram Fx.

**CutDirection****Enum**

## Description

Sets/Queries the direction of the cut the persistence histogram Fx.

## Values

<b>Horizontal</b>	Cut persistence map horizontally
<b>Vertical</b>	Cut persistence map vertically

**PART TWO: REFERENCE**

---

**HorCutCenter****Double**

Range: From -10 to 10, step 0.0001

## Description

Sets/Queries the position of the center of the horizontal cut (measured in the units of vertical scale) of the slice to be used in the persistence histogram Fx.

**HorCutWidth****Double**

Range: From -10 to 10, step 0.0001

## Description

Sets/Queries the width of the horizontal cut (measured in the units of vertical scale) of the slice to be used in persistence histogram Fx.

**PctCutWidth****Double**

Range: From 0 to 100, step 0.1

## Description

Sets/Queries the width of the cut in percent.

**VerCutCenter****Double**

Range: From -1 to 1, step 1e-009

## Description

Sets/Queries the position of the center of the vertical cut (measured in the units of horizontal scale) of the slice to be used in persistence histogram Fx.

**VerCutWidth****Double**

Range: From -1 to 1, step 1e-009

## Description

Sets/Queries the width of the vertical cut (measured in the units of horizontal scale) of the slice to be used in the persistence histogram Fx.

## PTRACE MEAN *app.Math.Fx.OperatorYSetup (Operator = "PersistenceTraceMean")*

Create a waveform from the mean of a persistence map.

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to generate a waveform from the
' mean value of the persistence map of C1
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Ptrace mean"
app.Math.F1.Operator1Setup.ClearSweeps
```

### ClearSweeps

### Action

#### Description

Clears the contents of persistence mean trace Fx.

## PART TWO: REFERENCE

**PTRACE RANGE** *app.Math.Fx.OperatorYSetup (Operator = "PersistenceTraceRange")*

Generates a waveform with a width derived from a population range of a persistence map.

<b>ClearSweeps</b>	Action
<b>PctPopulation</b>	Double

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Ptrace range"
app.Math.F1.Operator1Setup.ClearSweeps
app.Math.F1.Operator1Setup.PctPopulation = 50.0
```

**ClearSweeps****Action**

## Description

Clears the contents of persistence range trace Fx.

**PctPopulation****Double**

Range: From 0.5 to 100, step 0.5

## Description

Sets/Queries the percentage of the persistence population that is spanned by persistence range trace Fx.

**PTRACE SIGMA***app.Math.Fx.OperatorYSetup (Operator = "PersistenceTraceSigma")*

Generates a waveform with a width derived from the sigma of a persistence map.

<b>ClearSweeps</b>	Action
<b>Sigma</b>	Double

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Ptrace sigma"
app.Math.F1.Operator1Setup.ClearSweeps
app.Math.F1.Operator1Setup.Sigma = 5.0
```

**ClearSweeps****Action**

Description

Clears the contents of persistence sigma trace Fx.

**Sigma****Double**

Range: From 0.5 to 10, step 0.1

Description

Sets/Queries the number of standard deviations of the persistence population that is spanned by sigma trace Fx.

## PART TWO: REFERENCE

## RESCALE

*app.Math.Fx.OperatorYSetup (Operator = "Rescale")*

Linearly transform the vertical values of a waveform.

<b>Adder</b>	Double
<b>CustomUnit</b>	Bool
<b>Multiplier</b>	Double
<b>Unit</b>	String

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Rescale"
app.Math.F1.Operator1Setup.Adder = 2.0
app.Math.F1.Operator1Setup.Multiplier = 3.0
app.Math.F1.Operator1Setup.CustomUnit = True
app.Math.F1.Operator1Setup.Unit = "DEG"
```

**Adder****Double**

Range: From -1.79769e+308 to 1.79769e+308, step 0

Description

Sets/Queries the additive constant A in the rescale function  $Fx = M.Input + A$

**CustomUnit****Bool**

Description

Enables/Disables the application of a custom unit of measurement to rescale function trace Fx.

**Multiplier****Double**

Range: From -1.79769e+308 to 1.79769e+308 step 0

Description

Sets/Queries the multiplicative constant M in the rescale function  $Fx = M.Input + A$ .

**Unit*****String***

Range: Any number of characters

## Description

Sets/Queries the custom unit for rescale function trace Fx. Only used when the CustomUnit control is set to True.

## PART TWO: REFERENCE

## ROOF

*app.Math.Fx.OperatorYSetup (Operator = "Roof")*

The most positive or maximum values for an ensemble of sweeps, or "Roof."

<b>ClearSweeps</b>	Action
<b>Sweeps</b>	Integer

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to measure the Roof of the first 1000
' sweeps of C1
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Roof"
app.Math.F1.Operator1Setup.Sweeps = 1000
```

**ClearSweeps****Action**

## Description

Initiates a clear sweeps action for roof function trace Fx.

**Sweeps****Integer**

Range: From 1 to 1000000, step 1

## Description

Sets/Queries the maximum number of sweeps for Roof function trace Fx.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set function trace F2 to roof.
app.Math.F2.Operator1 = "Roof"
' Set the maximum number of sweeps to 150.
app.Math.F2.Operator1Setup.Sweeps = 150
```

**SEGMENT** *app.Math.Fx.OperatorYSetup (Operator = "SegmentSelect")*

Selects one waveform from a group of waveforms.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Enable sequence acquisition mode, collect 10 segments
app.Acquisition.Horizontal.NumSegments = 10
app.Acquisition.Horizontal.SampleMode = "Sequence"

' Configure F1 to show the 5th of the 10 collected segments
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Segment"
app.Math.F1.Operator1Setup.SelectedSegment = 5
```

**SelectedSegment***Integer*

Range: From 1 to 1000000000, step 1

**Description**

In sequence mode, sets/queries the number of the segment selected from a set.

## PART TWO: REFERENCE

**SPARSE***app.Math.Fx.OperatorYSetup (Operator = "Sparse")*

Waveform sparser, will reduce the number of points in the output waveform by skipping points in the input, and starting at a given offset.

<b>SparsingFactor</b>	Integer
<b>SparsingPhase</b>	Integer

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Configure F1 to sparse C1 by a factor of 100
app.Math.F1.View = True
app.Math.F1.Source1 = "C1"
app.Math.F1.MathMode = "OneOperator"
app.Math.F1.Operator1 = "Sparse"
app.Math.F1.Operator1Setup.SparsingFactor = 100
```

**SparsingFactor***Integer*

Range: From 1 to 1000000, step 1

## Description

Sets/Queries the factor by which the number of samples is reduced in the sparsing function Fx.

**SparsingPhase***Integer*

Range: From 0 to 0, step 1

## Description

Sets/Queries the number of the first sample that will be accepted by the sparsing function Fx.

**TRACK***app.Math.Fx.OperatorYSetup (Operator = "Track")*

The "Track" function gives a waveform of equivalent horizontal scale to the source waveform, but of a measurement on that waveform.

<b>AutoFindScale</b>	Bool
<b>Center</b>	Double
<b>FindScale</b>	Action
<b>VerScale</b>	DoubleLockstep

**AutoFindScale*****Bool***

Description

Enables/Disables the automatic setting of the vertical scale and vertical offset for Track trace Fx.

**Center*****Double***Range: From  $-1e+010$  to  $1e+010$ , step  $1e-012$ 

Description

Sets/Queries the vertical position of the center of Track trace Fx.

**FindScale*****Action***

Description

Sets the vertical scale and offset to optimum values to display Track trace Fx.

**VerScale*****DoubleLock step***Range: From  $1e-012$  to  $1e+012$  step 0.01, locked to 1-2-5

Description

Sets/Queries the vertical scale of Track trace Fx.

## PART TWO: REFERENCE

**TREND***app.Math.Fx.OperatorYSetup (Operator = "Trend")*

Trend of the values of a parameter, if connected to a parameter result source, or a trend of the sample values of a waveform, if connected to a waveform result.

<b>AutoFindScale</b>	Bool
<b>Center</b>	Double
<b>ClearSweeps</b>	Action
<b>FindScale</b>	Action
<b>Values</b>	Integer
<b>VerScale</b>	DoubleLockstep

**AutoFindScale****Bool**

Description

Enables/Disables the automatic setting of the vertical scale and vertical offset for Trend trace Fx.

**Center****Double**

Range: From -1e+010 to 1e+010, step 1e-012

Description

Sets/Queries the vertical position of the center of Trend trace Fx.

**ClearSweeps****Action**

Description

Clears the contents of Trend trace Fx.

**FindScale****Action**

Description

Sets the vertical scale and offset to optimum values to display Trend trace Fx.

**Values****Integer**

Range: From 20 to 1000000, step 1

Description

Sets/Queries the number of visible values in Trend trace Fx.

**VerScale****DoubleLock step**

Range: From 1e-012 to 1e+012 step 0.01, locked to 1-2-5

Description

Sets/Queries the vertical scale of Trend trace Fx.

**MATHCADPARAMARITH** *app.Measure.Px.Operator (ArithEngine = "MathcadParamArith")*

<b>Advanced</b>	Bool
<b>NewSheet</b>	Bool
<b>OutputEnable</b>	Bool
<b>OutputHeaderVar</b>	String
<b>OutputVar</b>	String
<b>Reload</b>	Action
<b>Source1Enable</b>	Bool
<b>Source1HeaderVar</b>	String
<b>Source1Var</b>	String
<b>Source2Enable</b>	Bool
<b>Source2HeaderVar</b>	String
<b>Source2Var</b>	String
<b>Status</b>	String
<b>WithHeader</b>	Bool
<b>WorksheetFilename</b>	FileName

**Advanced****Bool**

## Description

Enables/Disables/Queries use of advanced features for Mathcad parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Enables use of advanced features.
app.Measure.P3.Operator.Advanced = True
```

**NewSheet****Bool**

## Description

Enables/Disables/Queries the use of a new Mathcad worksheet for parameters.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Enable new Mathcad worksheet.
app.Measure.P3.Operator.NewSheet = True
```

**PART TWO: REFERENCE**

---

**OutputEnable****Bool**

## Description

Enables/Disables/Queries transmission of output data from Mathcad to the instrument.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Enables transmission of Mathcad output to the instrument.
app.Measure.P3.Operator.OutputEnable = True
```

**OutputHeaderVar****String**

Range: Any number of characters

## Description

Sets/Queries the name of the output variable header in Mathcad parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Sets the name of the Mathcad output header variable
app.Measure.P3.Operator.OutputHeaderVar = "outputheader"
```

**OutputVar****String**

Range: Any number of characters

## Description

Sets/Queries the name of the output variable in Mathcad parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Sets the name of the Mathcad output variable
app.Measure.P3.Operator.OutputVar = "output7"
```

**Reload****Action**

## Description

Reloads the specified Mathcad file.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Specify a Mathcad worksheet filename.
app.Measure.P3.Operator.WorksheetFilename =
"C:\XStreamMathcad\Param233.mcd"
' Reload the specified Mathcad file.
app.Measure.P3.Operator.Reload
```

**Source1Enable****Bool**

## Description

Enables/Disables/Queries transmission of source data 1 from instrument to Mathcad.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Enables transmission of source 1 data to instrument.
app.Measure.P3.Operator.Source1Enable = True
```

**Source1HeaderVar****String**

Range: Any number of characters

## Description

Sets/Queries the name of the input variable 1 header in Mathcad parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Sets the name of the Mathcad source 1 header variable
app.Measure.P3.Operator.Source1HeaderVar = "input1header"
```

## PART TWO: REFERENCE

---

### Source1Var

**String**

Range: Any number of characters

Description

Sets/Queries the name of input variable 1 in Mathcad parameter Px.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Sets the name of the source 1 variable
app.Measure.P3.Operator.Source1Var = "input1"
```

### Source2Enable

**Bool**

Description

Enables/Disables/Queries transmission of source data 2 from instrument to Mathcad.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Enables transmission of source 2 data to instrument.
app.Measure.P3.Operator.Source2Enable = True
```

### Source2HeaderVar

**String**

Range: Any number of characters

Description

Sets/Queries the name of input variable 2 header in Mathcad parameter Px.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Sets the name of the Mathcad source 2 header variable
app.Measure.P3.Operator.Source2HeaderVar = "input2header"
```

**Source2Var****String**

Range: Any number of characters

## Description

Sets/Queries the name of input variable 2 in Mathcad parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Sets the name of the source 2 variable
app.Measure.P3.Operator.Source2Var = "input2"
```

**Status****String**

Range: Any number of characters

## Description

Inspects the status of Mathcad parameter calculation Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Inspect status of Mathcad parameter calculation.
MathcadStatus = app.Measure.P3.Operator.Status
```

**WithHeader****Bool**

## Description

Enables/Disables/Queries the presence of headers with variables with Mathcad parameter Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Enables inclusion of headers with data.
app.Measure.P3.Operator.WithHeader = True
```

**PART TWO: REFERENCE**

---

**WorksheetFilename****FileName**

Range: Any number of characters

Description

Sets/Queries the name of the current Mathcad file for parameter Px.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 as Mathcad calculation.
app.Measure.P3.ParamEngine = "MathcadParam"
' Specify a Mathcad worksheet filename.
app.Measure.P3.Operator.WorksheetFilename =
"C:\XStreamMathcad\Param233.mcd"
```

**P INVERT***app.Measure.Px.Operator (ArithEngine = "ParamInvert")*

This engine produces the reciprocal of the input parameter.

**CycleForTimeUnits****Bool**

## Description

Sets/Queries the use of period as the measurement for time units, rather than 1/Hz.

When true, if there are seconds in the vertical dimensions of the source parameter, it is assumed that there is an implicit "cycles" per second. Thus the resulting units upon inversion is Hertz (cycles per second).

## PART TWO: REFERENCE

**P SCRIPT***app.Measure.Px.Operator (ArithEngine = "ParamScript")*

Calculates a parameter result from one or two parameter inputs

<b>Code</b>	String
<b>Language</b>	Enum
<b>Status</b>	String
<b>Timeout</b>	Double

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Place sample program code in a string
' This will the square of the input value times 3.1
code = "function Update()" + vbcrLf
code = code + "    value = InResult.Value(False)" + vbcrLf
code = code + "    OutResult.Value(False) = value * value * 3.1"
+ vbcrLf
code = code + "end Function"

' Configure P1 as a scripting component
app.Measure.P1.View = True
app.Measure.P1.Source1 = "C1"
app.Measure.P1.MeasurementType = "math"
app.Measure.P1.ArithEngine = "P Script"
app.Measure.P1.Operator.Language = "VBScript"
app.Measure.P1.Operator.Code = code
```

**Code****String**

Range: Any number of characters

## Description

Sets/Queries the code used to calculate parameter Px. This code is, of course, text.

**Language****Enum**

## Description

Sets/Queries the language for parameter script Fx (i.e., choice of scripting language).

## Values

JScript  
VBScript

**Status***String*

Range: Any number of characters

## Description

Inspects the status of parameter script Px. A typical message is "Error in line 14, Expected end of statement."

**Timeout***Double*

Range: From 1 to 12000, step 0.001

## Description

Sets/Queries the time-out for parameter script Px.

**PART TWO: REFERENCE**

---

**Q** *app.Measure.Px.Operator (ParamEngine = "EyeQ")*

Calculates the Q-factor based on a vertical slice from the (assumed input) eye diagram. The width of the vertical slice is specified through PctCutWidth

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.Pl.ParamEngine = "EyeQ"
```

**PctCutWidth****Double**

Range: From 0 to 100, step 0.1

**Description**

Sets/Queries the vertical slice or cut width in percent.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Sets the cut width to 8%.
app.Measure.Pl.Operator.PctCutWidth = 8
```

## AREA

*app.Measure.Px.Operator (ParamEngine = "Area")*

Calculates the area of the input waveform relative to zero.

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to area.
app.Measure.P1.View = True
app.Measure.P1.MeasurementType = "measure"
app.Measure.P1.ParamEngine = "Area"
app.Measure.P1.Source1 = "C1"

' Enable cyclic calculation of area.
app.Measure.P1.Operator.Cyclic = True

' Set markers to simple.
app.Measure.P1.Operator.CursorDisplay = "Simple"
```

### Cyclic

**Bool**

#### Description

Enables/Disables cyclic calculation of area parameter Px, which is calculated using a whole number of cycles of the signal.

**Note:** Help Markers aid in selecting the region of the waveform where measurement is made.

## PART TWO: REFERENCE

---

### **AVG POWER** *app.Measure.Px.Operator (ParamEngine = "EyeAvgPower")*

The measurement is assumed to be on an eye-diagram, and the percentage of a unit-interval at the center of the eye is used to estimate the average power.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

app.Measure.P1.ParamEngine = "EyeAvgPower"

' Sets the cut width to 8% of the unit interval.
app.Measure.P1.Operator.PctCutwidth = 8
```

#### **PctCutWidth**

**Double**

Range: From 0 to 100, step 0.1

#### Description

Sets/Queries the cut width in percent.

**DPERIOD@LEVEL***app.Measure.Px.Operator (ParamEngine = "DeltaPeriodAtLevel")*

Delta-period at level, the difference between successive (adjacent periods), also known as "cycle-to-cycle jitter."

<b>AbsLevel</b>	Double
<b>BaseFrequency</b>	Double
<b>FindBaseFrequency</b>	Action
<b>FindLevel</b>	Action
<b>GroupSize</b>	Integer
<b>Hysteresis</b>	Double
<b>LevelType</b>	Enum
<b>PercentLevel</b>	Double
<b>SignalType</b>	Enum
<b>Slope</b>	Enum
<b>StartCycle</b>	Integer
<b>StdBaseFrequency</b>	Enum
<b>UseBaseFrequency</b>	Enum

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.MeasureMode = "MyMeasure"
app.Measure.P1.MeasurementType = "measure"
app.Measure.P1.ParamEngine = "DeltaPeriodAtLevel"
' Set level type to absolute.
app.Measure.P1.Operator.LevelType = "Absolute"
' Set the measurement level to 5 mV.
app.Measure.P1.Operator.AbsLevel = 0.005
```

**AbsLevel****Double**

Range: From -100 to 100, step 1e-007

**Description**

Sets/Queries the absolute measuring level for parameter period at level Px.

## PART TWO: REFERENCE

---

### BaseFrequency

**Double**

Range: From 1 to 4e+010, step 1000

#### Description

Sets/Queries the base frequency of the signal for the parameter delta period at level. This only has effect when the UseBaseFrequency = "Custom", otherwise the base frequency is set directly from the standard selection.

When the period measurements are made on a data-stream instead of a clock, the engine needs to be told, or to learn, the base frequency (effective clock frequency). Since precision is exceptionally important, it's most often the case that the engine is asked to find the frequency, and then this control is queried to verify that the correct frequency has been found.

### FindBaseFrequency

**Action**

#### Description

Finds the base frequency of the signal by inspecting the signal, if you are not supplying a standard value or a custom value.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.P1.ParamEngine = "DeltaPeriodAtLevel"
' Find the base frequency of the signal.
app.Measure.P1.Operator.FindBaseFrequency
```

### FindLevel

**Action**

#### Description

Finds the 50 percent level for delta period measurements. Level type may be either absolute or percentage.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.P1.ParamEngine = "DeltaPeriodAtLevel"
' Find the 50 percent level.
app.Measure.P1.Operator.FindLevel
```

**GroupSize***Integer*

Range: From 1 to 128, step 1

## Description

Sets/Queries the size of a group of consecutive cycles for the delta period at level parameter Px. This is an advanced feature.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.P1.ParamEngine = "DeltaPeriodAtLevel"
' Set up for clock signal.
app.Measure.P1.Operator.SignalType = "Clock"
' Set the group size to 8.
app.Measure.P1.Operator.GroupSize = 8
```

**Hysteresis***Double*

Range: From 0 to 10, step 0.1

## Description

Sets the hysteresis range for parameter delta-period at level Px, specified in divisions. Hysteresis can be used to obtain good noise rejection.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.P1.ParamEngine = "DeltaPeriodAtLevel"
' Set the hysteresis range to 0.9 division. (+- 0.45 divisions)
app.Measure.P1.Operator.Hysteresis = 0.9
```

**LevelType***Enum*

## Description

Sets/Queries whether the level is absolute or a percentage of the amplitude of the signal for Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.P1.ParamEngine = "DeltaPeriodAtLevel"
```

## PART TWO: REFERENCE

---

```
' Set level type to absolute.
app.Measure.Pl.Operator.LevelType = "Absolute"
```

### Values

```
Absolute
Percent
```

### PercentLevel

**Double**

Range: From 0 to 100, step 1

### Description

Sets/Queries the percentage measuring level for parameter delta-period at level Px.

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.Pl.ParamEngine = "DeltaPeriodAtLevel"
' Set level type to percentage.
app.Measure.Pl.Operator.LevelType = Percent
' Set the measurement level to 55%
app.Measure.Pl.Operator.PercentLevel = 55
```

### SignalType

**Enum**

### Description

Sets/Queries the type of data specified for the measurement of delta-period at level.

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.Pl.ParamEngine = "DeltaPeriodAtLevel"
' Set the signal type to data.
app.Measure.Pl.Operator.SignalType = "Data"
```

### Values

```
Clock
Data
```

**Slope****Enum**

## Description

Sets/Queries the polarity of the transitions for measuring parameter delta-period at level.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.P1.ParamEngine = "DeltaPeriodAtLevel"
' Set transtion polarity to negative.
app.Measure.P1.Operator.LevelType = "Neg"
```

## Values

Both  
Neg  
Pos

## PART TWO: REFERENCE

---

### StartCycle

*Integer*

Range: From 0 to 0, step 1

#### Description

Sets/Queries the start cycle when measuring over groups of clock cycles

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.P1.ParamEngine = "DeltaPeriodAtLevel"
' Set up for clock signal.
app.Measure.P1.Operator.SignalType = "Clock"
' Set the group size to 8.
app.Measure.P1.Operator.GroupSize = 8
' Set the start cycle to 2.
app.Measure.P1.Operator.StartCycle = 2
```

### StdBaseFrequency

*Enum*

#### Description

Sets/Queries the standard base frequency, sending or receiving a string from the list of standard frequencies.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.P1.ParamEngine = "DeltaPeriodAtLevel"
' Set up to use the standard frequency
app.Measure.P1.Operator.UseBaseFrequency = "Standard"
' Set the standard base frequency to 2.048 MHz.
app.Measure.P1.Operator.StdBaseFrequency = "2.048MHz"
```

#### Values

1.544MHz  
139.264MHz  
155.52MHz  
2.048MHz  
2488.32MHz  
34.368MHz  
44.736MHz

51.84MHz  
622.08MHz  
8.448MHz

**UseBaseFrequency****Enum**

## Description

Sets/Queries the choice of a specified base frequency as custom or standard. When "custom" is selected, you must specify or find the base frequency.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta period at level.
app.Measure.P1.ParamEngine = "DeltaPeriodAtLevel"
' Set up to use a standard base frequency.
app.Measure.P1.Operator.UseBaseFrequency = "Standard"
```

## Values

Custom  
Standard

## PART TWO: REFERENCE

**DTIME@LEVEL***app.Measure.Px.Operator (ParamEngine = "DeltaTimeAtLevel")*

A calculation of the time between two highly specific transitions.

<b>AbsLevel1</b>	Double
<b>AbsLevel2</b>	Double
<b>FindLevel1</b>	Action
<b>FindLevel2</b>	Action
<b>Hysteresis1</b>	Double
<b>Hysteresis2</b>	Double
<b>LevelType1</b>	Enum
<b>LevelType2</b>	Enum
<b>PercentLevel1</b>	Double
<b>PercentLevel2</b>	Double
<b>Slope1</b>	Enum
<b>Slope2</b>	Enum

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.MeasureMode = "MyMeasure"
app.Measure.P1.MeasurementType = "measure"
app.Measure.P1.ParamEngine = "DeltaTimeAtLevel"
' Set level type for trace 1 to absolute.
app.Measure.P1.Operator.LevelType1 = "Absolute"
' Set the measurement level for trace 1 to 25 mV
app.Measure.P1.Operator.AbsLevel1 = 0.025
```

**AbsLevel1****Double**

Range: From -100 to 100, step 1e-005

**Description**

Sets/Queries the absolute measuring level for the first transition time, if LevelType1 = "Absolute".

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "DeltaTimeAtLevel"
' Set level type for trace 1 to absolute.
app.Measure.P1.Operator.LevelType1 = "Absolute"
' Set the measurement level for trace 1 to 25 mV
app.Measure.P1.Operator.AbsLevel1 = 0.025
```

**AbsLevel2****Double**

Range: From -100 to 100, step 1e-005

## Description

Sets/Queries the absolute measuring level for the second transition time, if LevelType2 = "Absolute".

**FindLevel1****Action**

## Description

Sets the measurement level automatically for the first transition time. (Works for both percent and absolute level modes.)

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "DeltaTimeAtLevel"
' Find the level for trace 1.
app.Measure.P1.Operator.FindLevel1
```

**FindLevel2****Action**

## Description

Sets the measurement level automatically for second transition (for LevelType2 either "Percent" or "Absolute").

**Hysteresis1****Double**

Range: From 0 to 10, step 0.1

## Description

Sets/Queries the hysteresis level in graticule divisions for first transition time. (This is an advanced feature.)

**Hysteresis2****Double**

Range: From 0 to 10, step 0.1

## Description

Sets/Queries the hysteresis level in graticule divisions for the second transition. (This is an advanced feature.)

**PART TWO: REFERENCE**

---

**LevelType1****Enum**

## Description

Sets/Queries whether to use absolute level or percent level for the first transition time.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "DeltaTimeAtLevel"
' Set level type for trace 1 to percentage.
app.Measure.P1.Operator.LevelType1 = "Percent"
' Set the measurement level for trace 1 to 55%
app.Measure.P1.Operator.PercentLevel1 = 55
```

## Values

Absolute  
Percent

**LevelType2****Enum**

## Description

Sets/Queries whether to use absolute level or percent level for specifying the second transition.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "DeltaTimeAtLevel"
' Set level type for trace 2 to absolute.
app.Measure.P1.Operator.LevelType2 = "Absolute"
' Set the measurement level for trace 2 to 15 mV
app.Measure.P1.Operator.AbsLevel2 = 0.015
```

## Values

Absolute  
Percent

**PercentLevel1****Double**

Range: From 0 to 100, step 1

## Description

Sets/Queries the measuring level, in percent, for the first transition if LevelType1 = "Percent".

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "DeltaTimeAtLevel"
' Set level type for trace 1 to percentage.
app.Measure.P1.Operator.LevelType1 = "Percent"
' Set the measurement level for trace 1 to 55%
app.Measure.P1.Operator.PercentLevel1 = 55
```

**PercentLevel2****Double**

Range: From 0 to 100, step 1

**Description**

Sets/Queries the measuring level, in percent, for the second transition if LevelType2 = "Percent".

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "DeltaTimeAtLevel"
' Set level type for trace 2 to percentage.
app.Measure.P1.Operator.LevelType2 = "Percent"
' Set the measurement level for trace 2 to 45%
app.Measure.P1.Operator.PercentLevel2 = 45
```

**Slope1****Enum****Description**

Sets/Queries the transition polarity of trace 1 for delta time at level Fx.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "DeltaTimeAtLevel"
' Set edge polarity for trace 1 to positive.
app.Measure.P1.Operator.Slope1 = "Pos"
```

**Values**

Both  
Neg  
Pos

## PART TWO: REFERENCE

---

### Slope2

*Enum*

#### Description

Sets/Queries the transition polarity of the second transition time.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "DeltaTimeAtLevel"
' Set edge polarity for the second transition to both.
app.Measure.P1.Operator.Slope2 = "Both"
```

#### Values

Both  
Neg  
Pos

**DUTY@LEVEL***app.Measure.Px.Operator (ParamEngine = "DutyAtLevel")*

<b>AbsLevel</b>	Double
<b>FindLevel</b>	Action
<b>Hysteresis</b>	Double
<b>HysteresisPct</b>	Double
<b>HysteresisType</b>	Enum
<b>LevelType</b>	Enum
<b>PercentLevel</b>	Double
<b>Slope</b>	Enum

**AbsLevel****Double**

Range: From -100 to 100, step 1e-007

Description

Sets/Queries the absolute level used if LevelType = "Absolute"

**FindLevel****Action**

Description

Causes the engine to find a suitable level for either LevelType ("Absolute" or "Percent").

**Hysteresis****Double**

Range: From 0 to 10, step 0.1

Description

Sets/Queries the hysteresis range in divisions for duty at level Px.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "DutyAtLevel"
' Set hysteresis measurement type to divisions.
app.Measure.P1.Operator.HysteresisType = "Divisions"
' Set the hysteresis to 0.8 divisions.
app.Measure.P1.Operator.Hysteresis = 0.8
```

## PART TWO: REFERENCE

---

### HysteresisPct

*Double*

Range: From 0 to 100, step 0.1

#### Description

Sets/Queries the percent hysteresis range for duty at level Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "DutyAtLevel"
' Set hysteresis measurement type to percentage.
app.Measure.P1.Operator.HysteresisType = "Percent"
' Set the percent hysteresis to 55%
app.Measure.P1.Operator.HysteresisPct = 55
```

### HysteresisType

*Enum*

#### Description

Sets/Queries whether to set the hysteresis range in divisions or percent of the signal amplitude.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "DutyAtLevel"
' Set hysteresis measurement type to percentage.
app.Measure.P1.Operator.HysteresisType = "Percent"
' Set the percent hysteresis to 55%
app.Measure.P1.Operator.HysteresisPct = 55
```

#### Values

Divisions  
Percent

### LevelType

*Enum*

#### Values

Absolute  
Percent

**PercentLevel***Double*

Range: From 0 to 100, step 1

## Description

Using ParamEngine = "DutyAtLevel", refer to the corresponding variable for the Delta Time At Level parameter.

**Slope***Enum*

## Description

Sets/Queries the polarity of the first transition used (i.e., classic duty factor is for setting "Pos")

## Values

Neg  
Pos

## PART TWO: REFERENCE

**DWIDTH@LEVEL***app.Measure.Px.Operator (ParamEngine = "DeltaWidthAtLevel")*

Calculates the Difference (delta) between adjacent width measurements.

<b>AbsLevel</b>	Double
<b>FindLevel</b>	Action
<b>Hysteresis</b>	Double
<b>LevelType</b>	Enum
<b>PercentLevel</b>	Double
<b>Slope</b>	Enum

**AbsLevel****Double**

Range: From -100 to 100, step 1e-007

Description

Sets/Queries the absolute level used if LevelType = "Absolute"

**FindLevel****Action**

Description

Causes the engine to find a suitable level for either LevelType ("Absolute" or "Percent")

**Hysteresis****Double**

Range: From 0 to 10, step 0.1

Description

Sets/Queries the hysteresis setting for this measurement. (This is an advanced feature.)

**LevelType****Enum**

Description

Sets/Queries whether to use absolute level or percent level for the last (ending) transition time.

Values

Absolute  
Percent**PercentLevel****Double**

Range: From 0 to 100, step 1

Description

Sets/Queries the percentage level to be used if the LevelType = "Percent".

**Slope****Enum**

Description

Sets/Queries which polarity transition is used as the start (first) of the width measurement.

Values

Both

Neg

Pos

## PART TWO: REFERENCE

**EDGE@LEVEL***app.Measure.Px.Operator (ParamEngine = "EdgeAtLevel")*

Counts the number of transitions or edges in the input waveform.

<b>AbsLevel</b>	Double
<b>FindLevel</b>	Action
<b>Hysteresis</b>	Double
<b>LevelType</b>	Enum
<b>PercentLevel</b>	Double
<b>Slope</b>	Enum

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

app.Measure.P1.ParamEngine = "EdgeAtlevel"
```

**AbsLevel****Double**

Range: From -100 to 100, step 1e-007

## Description

Sets/Queries the absolute level used to detect edges, when LevelType = "Absolute"

**FindLevel****Action**

## Description

Causes the engine to find a suitable level for either LevelType ("Absolute" or "Percent")

**Hysteresis****Double**

Range: From 0 to 10, step 0.1

## Description

Sets/Queries the hysteresis used for edge detection in divisions. (This is an advanced feature.)

**LevelType****Enum**

## Description

Sets/Queries whether to use absolute or percent levels.

## Values

Absolute

Percent

**PercentLevel****Double**

Range: From 0 to 100, step 1

## Description

Sets/Queries the percent level threshold used for detection of edges when LevelType = "Percent".

**Slope***Enum*

## Description

Sets/Queries the polarity of the edges detected.

## Values

Both

Neg

Pos

## PART TWO: REFERENCE

## EXCELPARAM

*app.Measure.Px.Operator (ParamEngine = "ExcelParam")*

<b>AddChart</b>	Action
<b>AddLabels</b>	Action
<b>Advanced</b>	Bool
<b>ClearSheet</b>	Action
<b>CreateDemoSheet</b>	Action
<b>NewSheet</b>	Bool
<b>OutputCell</b>	String
<b>OutputEnable</b>	Bool
<b>OutputHeaderCell</b>	String
<b>Source1Cell</b>	String
<b>Source1Enable</b>	Bool
<b>Source1HeaderCell</b>	String
<b>Source2Cell</b>	String
<b>Source2Enable</b>	Bool
<b>Source2HeaderCell</b>	String
<b>SpreadsheetFilename</b>	FileName
<b>Status</b>	String
<b>WithHeader</b>	Bool

**AddChart****Action**

Description

Adds a chart to the output worksheet.

**AddLabels****Action**

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**Advanced****Bool**

Description

Sets/Queries whether advanced features of this component are enabled.

**ClearSheet****Action**

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**CreateDemoSheet****Action**

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**NewSheet****Bool**

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**OutputCell****String**

Range: Any number of characters

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**OutputEnable****Bool**

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**OutputHeaderCell****String**

Range: Any number of characters

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**Source1Cell****String**

Range: Any number of characters

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**Source1Enable****Bool**

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**Source1HeaderCell****String**

Range: Any number of characters

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**Source2Cell****String**

Range: Any number of characters

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**Source2Enable****Bool**

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

## PART TWO: REFERENCE

---

### Source2HeaderCell

*String*

Range: Any number of characters

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

### SpreadsheetFilename

*FileName*

Range: Any number of characters

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

### Status

*String*

Range: Any number of characters

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

### WithHeader

*Bool*

Description

Using ParamEngine = "ExcelParam", refer to the corresponding variable for the ExcelMath function.

**EXT. RATIO***app.Measure.Px.Operator (ParamEngine = "ExtinctionRatio")*

Extinction Ratio assumes it is operating on an eye diagram (persistence result).

<b>CalcType</b>	Enum
<b>PctCutWidth</b>	Double

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to ExtinctionRatio
app.Measure.P1.ParamEngine = "ExtinctionRatio"
```

**CalcType****Enum****Description**

Sets/Queries the calculation type.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

app.Measure.P1.ParamEngine = "ExtinctionRatio"
' Set the calculation type to linear.
app.Measure.P1.Operator.CalcType = "Linear"
```

**Values**

dB  
linear

**PctCutWidth****Double**

Range: From 0 to 100, step 0.1

**Description**

Sets/Queries the percent cut width. This specifies which region of the eye diagram (about the center of the eye) will be used to estimate extinction ratio.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to delta time at level.
app.Measure.P1.ParamEngine = "ExtinctionRatio"
' Set the percent cut width to 15%.
app.Measure.P1.Operator.PctCutWidth = 15
```

## PART TWO: REFERENCE

---

### EYE AMPLITUDE

*app.Measure.Px.Operator (ParamEngine = "EyeAmplitude")*

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

app.Measure.P1.ParamEngine = "EyeAmplitude"
```

#### PctCutWidth

**Double**

Range: From 0 to 100, step 0.1

#### Description

Sets/Queries the percent cut width. This specifies which region of the eye diagram (about the center of the eye) will be used to estimate eye amplitude.

**EYE BER***app.Measure.Px.Operator (ParamEngine = "EyeBER")*

Estimates the BER based on a cut through the (assumed input) Eye Diagram.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.P1.ParamEngine = "EyeBER"
```

**PctCutWidth***Double*

Range: From 0 to 100, step 0.1

**Description**

Sets/Queries the percent cut width. This specifies which region of the eye diagram (about the center of the eye) will be used to estimate Bit Error Rate

## PART TWO: REFERENCE

**EYE HEIGHT**      *app.Measure.Px.Operator (ParamEngine = "EyeHeight")*

Estimates the difference between the high (one level) and low (zero level) of the (assumed input) eye diagram, based on a vertical slice centered on the eye diagram of a specified width, PctCutWidth.

<b>CalcUnits</b>	Enum
<b>PctCutWidth</b>	Double

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.Pl.ParamEngine = "EyeHeight"
app.Measure.Pl.Operator.PctCutWidth = 7
```

**CalcUnits****Enum**

## Description

Using ParamEngine = "EyeHeight", refer to the corresponding variable for the Extinction Ratio parameter.

## Values

dB  
linear

**PctCutWidth****Double**

Range:      From 0 to 100, step 0.1

## Description

Sets/Queries the percent cut width. This specifies which region of the eye diagram (about the center of the eye) will be used to estimate eye height.

**FALL@LEVEL** *app.Measure.Px.Operator (ParamEngine = "FallAtLevel")*

Calculates the fall time at specified arbitrary levels.

<b>HighAbs</b>	Double
<b>HighPct</b>	Double
<b>LevelsAre</b>	Enum
<b>LowAbs</b>	Double
<b>LowPct</b>	Double

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.P1.ParamEngine = "FallAtLevel"

' Set measurement level type to percent.
app.Measure.P1.Operator.LevelsAre = "Percent"
' Set the higher percent level to 85 percent.
app.Measure.P1.Operator.HighPct = 85
' Set the higher percent level to 15 percent.
app.Measure.P1.Operator.HighPct = 15
```

**HighAbs****Double**

Range: From -100 to 100, step 1e-007

**Description**

Sets/Queries the upper (first) transition level, if LevelsAre = "Absolute".

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to fall time at set levels.
app.Measure.P1.ParamEngine = "FallAtLevel"
' Set measurement level type to absolute.
app.Measure.P1.Operator.LevelsAre = "Absolute"
' Set the higher absolute level to 120 mV.
app.Measure.P1.Operator.HighAbs = 0.12
```

**HighPct****Double**

Range: From 11 to 95, step 1

**Description**

Sets/Queries the upper (first) transition level, if LevelsAre = "Percent".

**PART TWO: REFERENCE**

---

**LevelsAre****Enum**

## Description

Sets/Queries whether the measurement levels are absolute or relative to the trace.

## Values

Absolute

Percent

**LowAbs****Double**

Range: From -100 to 100, step 1e-007

## Description

Sets/Queries the lower (second) transition level, if LevelsAre = "Absolute".

**LowPct****Double**

Range: From 5 to 89, step 1

## Description

Sets/Queries the lower (second) transition level, if LevelsAre = "Percent".

**FREQ@LEVEL***app.Measure.Px.Operator (ParamEngine = "FrequencyAtLevel")*

Calculates the frequency based on a specified transition level.

<b>AbsLevel</b>	Double
<b>BaseFrequency</b>	Double
<b>FindBaseFrequency</b>	Action
<b>FindLevel</b>	Action
<b>Hysteresis</b>	Double
<b>LevelType</b>	Enum
<b>PercentLevel</b>	Double
<b>SignalType</b>	Enum
<b>Slope</b>	Enum
<b>StdBaseFrequency</b>	Enum
<b>UseBaseFrequency</b>	Enum

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.P1.ParamEngine = "FrequencyAtlevel"
```

**AbsLevel****Double**

Range: From -100 to 100, step 1e-007

**Description**

Sets/Queries the transition level, if LevelType = "Percent".

**BaseFrequency****Double**

Range: From 1 to 4e+010, step 1000

**Description**

Sets/Queries a custom (non-standard) base frequency to be used when SignalType = "Data" and when UseBaseFrequency = "Custom"

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to frequency at level.
app.Measure.P1.ParamEngine = "FrequencyAtLevel"

' Set the instrument to treat the input waveform as a data
stream.
app.Measure.P1.Operator.SignalType = "Data"

' Set the instrument to use a non-standard base frequency.
app.Measure.P1.Operator.UseBaseFrequency = "Custom"

' Set the base frequency to 23.79 MHz. (note this is a number not
```

## PART TWO: REFERENCE

---

```

    a string,
    'however a string would work just as well)
    app.Measure.Pl.Operator.BaseFrequency = 23.79e6

```

### FindBaseFrequency

**Action**

#### Description

Set the base frequency by inspection of the data signal, is SignalType = "Data"

#### Example

```

' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to frequency at level.
app.Measure.Pl.ParamEngine = "FrequencyAtLevel"
' Set the instrument to use a non-standard base frequency.
app.Measure.Pl.Operator.UseBaseFrequency = "Custom"
' Find the base frequency from the signal.
app.Measure.Pl.Operator.FindBaseFrequency

```

### FindLevel

**Action**

#### Description

Causes the engine to find a suitable level for either LevelType ("Absolute" or "Percent").

### Hysteresis

**Double**

Range: From 0 to 10, step 0.1

#### Description

Sets/Queries the hysteresis level. (This is an advanced feature.)

### LevelType

**Enum**

#### Description

Sets/Queries whether the levels used are specified in "Percent" or "Absolute".

#### Values

Absolute  
Percent

### PercentLevel

**Double**

Range: From 0 to 100, step 1

#### Description

Sets/Queries the transition level, if LevelType = "Percent".

**SignalType****Enum**

## Description

Sets/Queries the input signal type. When "Data" is selected, the periods are the same as for an NRZ datastream, with changing states on whole periods of the assumed "clock."

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to frequency at level.
app.Measure.P1.ParamEngine = "FrequencyAtLevel"
' Set input signal type to data
app.Measure.P1.Operator.SignalType = "Data"
```

## Values

Clock  
Data

**Slope****Enum**

## Description

Sets/Queries the polarity of the transitions to be used to estimate whole cycles.

## Values

Neg  
Pos

**StdBaseFrequency****Enum**

## Description

Sets/Queries the selection of standard frequencies (an enumerated list). This setting only has effect if UseBaseFrequency = "Standard".

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to frequency at level.
app.Measure.P1.ParamEngine = "FrequencyAtLevel"
' Set the instrument to use a standard base frequency.
app.Measure.P1.Operator.UseBaseFrequency = "Standard"
' Set the standard base frequency to 155.52MHz
app.Measure.P1.Operator.StdBaseFrequency = "155.52MHz"
'or alternately set to third choice in (zero based index) list
app.Measure.P1.Operator.StdBaseFrequency = 2
```

## PART TWO: REFERENCE

---

### Values

1.544MHz  
139.264MHz  
155.52MHz  
2.048MHz  
2488.32MHz  
34.368MHz  
44.736MHz  
51.84MHz  
622.08MHz  
8.448MHz

### UseBaseFrequency

### *Enum*

#### Description

Sets/Queries whether to use a custom base frequency or a standard base frequency for frequency at level.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to frequency at level.
app.Measure.P1.ParamEngine = "FrequencyAtLevel"
' Set the instrument to use a standard base frequency.
app.Measure.P1.Operator.UseBaseFrequency = "Standard"
```

#### Values

Custom  
Standard

**FWXX***app.Measure.Px.Operator (ParamEngine = "FullWidthAtXX")*

Calculates the full width (of a histogram or distribution) at a specified fraction of the maximum height. (This is a generalized case of FWHM, or full width at half maximum.)

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.P1.ParamEngine = "FullWidthAtXX"
'set the fraction to 25 percent
app.Measure.P1.Operator.HFractionHt = .25
```

**HFractionHt***Double*

Range: From 0 to 100, step 1

**Description**

Sets/Queries the fraction (as a percentage) of the maximum height at which the full width of a histogram will be measured.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Sets the fraction of the height for parameter P2 to 25%.
app.Measure.P2.Operator.HFractionHt = 25
```

## PART TWO: REFERENCE

**HALF PERIOD** *app.Measure.Px.Operator (ParamEngine = "HalfPeriod")*

AbsLevel	Double
FindLevel	Action
Hysteresis	Double
LevelType	Enum
PercentLevel	Double
Slope	Enum

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.Pl.ParamEngine = "HalfPeriod"
```

**AbsLevel****Double**

Range: From -100 to 100, step 1e-007

## Description

Sets/Queries the transition level, if LevelType = "Absolute".

**FindLevel****Action**

## Description

Causes the engine to find a suitable level for either LevelType ("Absolute" or "Percent").

**Hysteresis****Double**

Range: From 0 to 10, step 0.1

## Description

Sets/Queries the hysteresis level in divisions (this is an advanced feature).

**LevelType****Enum**

## Description

Sets/Queries whether the levels used are specified in "Percent" or "Absolute".

## Values

Absolute  
Percent

**PercentLevel****Double**

Range: From 0 to 100, step 1

## Description

Sets/Queries the transition level, if LevelType = "Percent".

**Slope***Enum*

## Description

Sets/Queries the polarity of the transitions to be used to estimate whole cycles.

## Values

Both

Neg

Pos

## PART TWO: REFERENCE

**HOLD TIME***app.Measure.Px.Operator (ParamEngine = "HoldTime")*

Calculates the "Hold" time for a data or clock signal.

<b>ClockAbsLevel</b>	Double
<b>ClockFindLevel</b>	Action
<b>ClockHysteresis</b>	Double
<b>ClockLevelIs</b>	Enum
<b>ClockPctLevel</b>	Double
<b>ClockSlope</b>	Enum
<b>CursorDisplay</b>	Enum
<b>DataAbsLevel</b>	Double
<b>DataFindLevel</b>	Action
<b>DataHysteresis</b>	Double
<b>DataLevelIs</b>	Enum
<b>DataPctLevel</b>	Double
<b>DataSlope</b>	Enum

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to hold time
app.Measure.P1.ParamEngine = "HoldTime"
' Set clock level type to absolute.
app.Measure.P1.Operator.ClockLevelIs = "Absolute"
' Set the absolute clock level to 30 mV.
app.Measure.P1.Operator.ClockAbsLevel = 0.03
'or alternatively to 32 millivolts
app.Measure.P1.Operator.ClockAbsLevel = "32mV"
```

**ClockAbsLevel****Double**

Range: From -100 to 100, step 1e-007

## Description

Sets/Queries the absolute measuring level for the clock input, if ClockLevelIs = "Absolute".

**ClockFindLevel****Action**

## Description

Causes the engine to find a suitable level for either ClockLevelIs setting ("Absolute" or "Percent").

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Set parameter P1 to hold time at set levels.
app.Measure.P1.ParamEngine = "HoldTime"

' Find the measuring level for the clock signal.
app.Measure.P1.Operator.ClockFindLevel
```

**ClockHysteresis***Double*

Range: From 0 to 10, step 0.1

## Description

Sets/Queries the hysteresis range in divisions for the clock input signal.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to hold time at set levels.
app.Measure.P1.ParamEngine = "HoldTime"

' Set the clock hysteresis to 0.8 divisions.
app.Measure.P1.Operator.ClockHysteresis = 0.8
```

**ClockLevelIs***Enum*

## Description

Sets/Queries whether the clock level is set as absolute or percent for hold time Px.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to hold time at set levels.
app.Measure.P1.ParamEngine = "HoldTime"

' Set clock level type to absolute.
app.Measure.P1.Operator.ClockLevelIs = "Absolute"
```

## Values

Absolute  
Percent

## PART TWO: REFERENCE

---

### ClockPctLevel

*Double*

Range: From 0 to 100, step 1

#### Description

Sets/Queries the percent measuring level for the clock input.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to hold time at set levels.
app.Measure.P1.ParamEngine = "HoldTime"
' Set clock level type to percent.
app.Measure.P1.Operator.ClockLevelIs = "Percent"
' Set the percent clock level to 45
app.Measure.P1.Operator.ClockPctLevel = 45
```

### ClockSlope

*Enum*

#### Description

Sets/Queries the slope for the measured transitions for the clock input signal.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to hold time at set levels.
app.Measure.P1.ParamEngine = "HoldTime"
' Set the clock slope to positive.
app.Measure.P1.Operator.ClockSlope = "Pos"
```

#### Values

Both  
Neg  
Pos

### CursorDisplay

*Enum*

#### Description

Sets/Queries the type of cursor display for hold time.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to hold time at set levels.
```

```

app.Measure.P1.ParamEngine = "HoldTime"
' Set the type of cursor display as simple.
app.Measure.P1.Operator.CursorDisplay = "Simple"

```

**Values**

```

Detailed
Off
Simple

```

**DataAbsLevel****Double**

Range: From -100 to 100, step 1e-007

**Description**

Sets/Queries the absolute measuring level for the data input.

**Example**

```

' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to hold time at set levels.
app.Measure.P1.ParamEngine = "HoldTime"
' Set data level type to absolute.
app.Measure.P1.Operator.DataLevelIs = "Absolute"
' Set the absolute data level to 30 mV.
app.Measure.P1.Operator.DataAbsLevel = 0.03

```

**DataFindLevel****Action****Description**

Automatically finds a suitable level for the Data signal, for either DataLevelIs ("Absolute" or "Percent").

**Example**

```

' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to hold time at set levels.
app.Measure.P1.ParamEngine = "HoldTime"
' Find the measuring level for the data signal.
app.Measure.P1.Operator.DataFindLevel

```

## PART TWO: REFERENCE

---

### DataHysteresis

*Double*

Range: From 0 to 10, step 0.1

#### Description

Sets/Queries the hysteresis range in divisions for the data input signal.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to hold time at set levels.
app.Measure.P1.ParamEngine = "HoldTime"
' Set the data hysteresis to 0.8 divisions.
app.Measure.P1.Operator.DataHysteresis = 0.8
```

### DataLevelIs

*Enum*

#### Description

Sets/Queries whether the data level is set as absolute or percent for hold time Px.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to hold time at set levels.
app.Measure.P1.ParamEngine = "HoldTime"
' Set data level type to percent.
app.Measure.P1.Operator.DataLevelIs = "Percent"
```

#### Values

Absolute  
Percent

### DataPctLevel

*Double*

Range: From 0 to 100, step 1

#### Description

Sets/Queries the percent measuring level for the data input.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to hold time at set levels.
app.Measure.P1.ParamEngine = "HoldTime"
' Set data level type to percent.
```

```
app.Measure.P1.Operator.DataLevelIs = "Percent"  
' Set the percent data level to 55  
app.Measure.P1.Operator.DataPctLevel = 55
```

**DataSlope****Enum**

## Description

Sets/Queries the polarity of the transitions measured for the data input signal.

## Example

```
' Visual Basic Script  
Set app = CreateObject("LeCroy.XStreamDSO")  
  
' Set parameter P1 to hold time at set levels.  
app.Measure.P1.ParamEngine = "HoldTime"  
' Set the data slope to negative.  
app.Measure.P1.Operator.DataSlope = "Neg"
```

## Values

Both  
Neg  
Pos

## PART TWO: REFERENCE

## LEVEL@X

*app.Measure.Px.Operator (ParamEngine = "LevelAtX")*

<b>CursorShape</b>	Enum
<b>HorValue</b>	Double
<b>LevelCursor</b>	Bool
<b>Marker</b>	Bool
<b>PinToData</b>	Bool
<b>TimeCursor</b>	Bool

**CursorShape***Enum*

## Description

Sets/Queries the cursor shape.

## Example

```
Set app = CreateObject("LeCroy.XStreamDSO")
```

```
' Set parameter P1 to level at X
app.Measure.P1.ParamEngine = "LevelAtX"
' Sets the cursor to difference.
app.Measure.P1.Operator.CursorShape = "Difference"
```

## Values

Absolute  
Difference  
Reference

**HorValue***Double*

Range: From -1.79769e+308 to 1.79769e+308, step 0

## Description

Sets/Queries the horizontal position at which the level is to be measured.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to level at X
app.Measure.P1.ParamEngine = "LevelAtX"
' Set the horizontal value to 120 ns.
app.Measure.P1.Operator.HorValue = 0.12e-6
```

**LevelCursor****Bool**

## Description

Sets/Queries whether the level cursor is visible.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to level at X
app.Measure.P1.ParamEngine = "LevelAtX"
' Remove the level cursor.
app.Measure.P1.Operator.LevelCursor = False
```

**Marker****Bool**

## Description

Sets/Queries whether the marker is visible.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to level at X
app.Measure.P1.ParamEngine = "LevelAtX"
' Remove the marker.
app.Measure.P1.Operator.Marker = False
```

**PinToData****Bool**

## Description

Sets/Queries whether the measurement is taken from the nearest data point (pin to data) or not (based on the interpolated value at X).

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to level at X
app.Measure.P1.ParamEngine = "LevelAtX"
' Disables pin to data to allow interpolated measurements.
app.Measure.P1.Operator.PinToData = False
```

## PART TWO: REFERENCE

---

### TimeCursor

**Bool**

#### Description

Sets/Queries whether the time cursor is visible.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to level at X
app.Measure.P1.ParamEngine = "LevelAtX"
' Remove the time cursor.
app.Measure.P1.Operator.TimeCursor = False
```

**MATHCADPARAM***app.Measure.Px.Operator (ParamEngine = "MathcadParam")*

<b>Advanced</b>	Bool
<b>NewSheet</b>	Bool
<b>OutputEnable</b>	Bool
<b>OutputHeaderVar</b>	String
<b>OutputVar</b>	String
<b>Reload</b>	Action
<b>Source1Enable</b>	Bool
<b>Source1HeaderVar</b>	String
<b>Source1Var</b>	String
<b>Source2Enable</b>	Bool
<b>Source2HeaderVar</b>	String
<b>Source2Var</b>	String
<b>Status</b>	String
<b>WithHeader</b>	Bool
<b>WorksheetFilename</b>	FileName

**Advanced****Bool**

Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**NewSheet****Bool**

Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**OutputEnable****Bool**

Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**OutputHeaderVar****String**

Range: Any number of characters

Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**OutputVar****String**

Range: Any number of characters

Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**PART TWO: REFERENCE**

---

**Reload****Action**

## Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**Source1Enable****Bool**

## Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**Source1HeaderVar****String**

Range: Any number of characters

## Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**Source1Var****String**

Range: Any number of characters

## Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**Source2Enable****Bool**

## Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**Source2HeaderVar****String**

Range: Any number of characters

## Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**Source2Var****String**

Range: Any number of characters

## Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**Status****String**

Range: Any number of characters

Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**WithHeader****Bool**

Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

**WorksheetFilename****FileName**

Range: Any number of characters

Description

Using ParamEngine = "MathcadParam", refer to the corresponding variable for the MathcadMath function.

## PART TWO: REFERENCE

**MATLAB PARAM***app.Measure.Px.Operator (ParamEngine = "MATLABParameter")*

<b>MATLABCode</b>	String
<b>MATLABPlot</b>	Bool
<b>MATLABScalePerDiv</b>	Double
<b>MATLABZeroOffset</b>	Double

**MATLABCode****String**

Range: Any number of characters

Description

Using ParamEngine = "MATLAB param", refer to the corresponding variable for the MATLABMath function.

**MATLABPlot****Bool**

Description

Using ParamEngine = "MATLAB param", refer to the corresponding variable for the MATLABMath function.

**MATLABScalePerDiv****Double**

Range: From 1e-009 to 1e+009, step 1e-009

Description

Using ParamEngine = "MATLAB param", refer to the corresponding variable for the MATLABMath function.

**MATLABZeroOffset****Double**

Range: From -1e+009 to 1e+009, step 1e-009

Description

Using ParamEngine = "MATLAB param", refer to the corresponding variable for the MATLABMath function.

**MAXIMUM***app.Measure.Px.Operator (ParamEngine = "Maximum")*

Calculates the maximum vertical value of the waveform.

## Example

```
' Visual Basic Script  
Set app = CreateObject("LeCroy.XStreamDSO")  
app.Measure.P1.ParamEngine = "Maximum"
```

## PART TWO: REFERENCE

---

### MEAN

*app.Measure.Px.Operator (ParamEngine = "Mean")*

Calculates the mean value of the input waveform's vertical values. When Cyclic = true, the range of values used is limited to a whole number of cycles.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to mean.
app.Measure.P1.ParamEngine = "Mean"

' Set the mean parameter for cyclic measurements.
app.Measure.P1.Operator.Cyclic = true
```

### Cyclic

**Bool**

#### Description

Sets/Queries whether the mean parameter Px is to be measured over a number of complete cycles.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P2 to mean.
app.Measure.P2.ParamEngine = "Mean"

' Set the mean parameter for cyclic measurements.
app.Measure.P2.ParamEngine.Cyclic = True
```

## MEDIAN

*app.Measure.Px.Operator (ParamEngine = "Median")*

Calculates the median (division between two halves) of the probability distribution of an input waveform. For periodic signals it is advisable to use Cyclic = True.

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.P1.ParamEngine = "Median"
' Set the measurement for a periodic signal
app.Measure.P1.Operator.Cyclic = true
```

## Cyclic

**Bool**

### Description

Sets/Queries whether the median parameter Px is to be measured over an integral number of complete cycles.

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P2 to median.
app.Measure.P2.ParamEngine = "Median"
' Set the median parameter for cyclic measurements.
app.Measure.P2.Operator.Cyclic = True
```

## PART TWO: REFERENCE

---

### MINIMUM

*app.Measure.Px.Operator (ParamEngine = "Minimum")*

Calculates the minimum value of a waveform.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

app.Measure.P1.ParamEngine = "Minimum"
```

**NB PHASE***app.Measure.Px.Operator (ParamEngine = "NarrowBandPhase")*

Estimates the phase of the frequency component at the specified Frequency.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to narrow band phase.
app.Measure.P1.ParamEngine = "NarrowBandPhase"
' Set the frequency at which the phase is to be measured to 691
MHz.
app.Measure.P1.Operator.Frequency = "691MHz"
' or alternatively to 299MHz
```

**Frequency****Double**

Range: From 10 to 1e+010, step 1

**Description**

Sets/Queries the frequency at which the narrow band phase is to be measured.

## PART TWO: REFERENCE

---

### NB POWER

*app.Measure.Px.Operator (ParamEngine = "NarrowBandPower")*

Measures the power found at a given frequency.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to narrow band phase.
app.Measure.P1.ParamEngine = "NarrowBandPower"
' Set the frequency at which the power is to be measured to 235
MHz.
```

#### Frequency

**Double**

Range: From 10 to 1e+010, step 1

#### Description

Sets/Queries the frequency at which the narrow band power is to be measured.

## NPTS

*app.Measure.Px.Operator (ParamEngine = "npoints")*

Counts the number of (sample or data) points in the input waveform.

### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.P3.ParamEngine = "npoints"

' Also count the points extending off the grid
app.Measure.P3.Operator.UsePointsInFrame = false
```

### UsePointsInFrame

**Bool**

#### Description

Sets/Queries whether to use the number of points in the graticule, or the total number of points in the trace.

**Note:** In many cases there are points off screen in the source waveform; but in particular, in the X-Stream standard architecture, waveforms are often "clipped to frame" already. This setting is most useful when using the WebEdit mode.

## PART TWO: REFERENCE

---

### ONE LEVEL

*app.Measure.Px.Operator (ParamEngine = "EyeOneLevel")*

Estimates the high level for the (assumed input) eye diagram, based on a slice through the middle of the eye diagram of specified width, PctCutWidth.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.Pl.ParamEngine = "EyeOneLevel"
app.Measure.Pl.Operator.PctCutWidth = 8
```

#### PctCutWidth

*Double*

Range: From 0 to 100, step 0.1

#### Description

Sets/Queries the width of the vertical slice through the eye, from which the one level is estimated.

**PARAM SCRIPT***app.Measure.Px.Operator (ParamEngine = "ParamScript")*

Calculates a parameter from a waveform based on VBScript or JavaScript.

<b>Code</b>	String
<b>Language</b>	Enum
<b>Status</b>	String
<b>Timeout</b>	Double

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Place sample program code in a string
' This will calculate the mean of all of the data points
code = "Function Update()" + vbcrLf
code = code + "    numSamples = InResult.Samples" + vbcrLf
code = code + "    unscaledData = InResult.DataArray(False)" +
vbcrLf
code = code + "    mean = 0" + vbcrLf
code = code + "    For i = 0 to numSamples-1" + vbcrLf
code = code + "        mean = mean + unscaledData(i)" + vbcrLf
code = code + "    next" + vbcrLf
code = code + "    if numSamples > 0 then mean = mean/numSamples"
+ vbcrLf
code = code + "    OutResult.Value = mean" + vbcrLf
code = code + "end Function"

' Configure P1 as a scripting component
app.Measure.P1.View = True
app.Measure.P1.Source1 = "C1"
app.Measure.P1.MeasurementType = "measure"
app.Measure.P1.ParamEngine = "ParamScript"
app.Measure.P1.Operator.Language = "VBScript"
```

**Code****String**

Range: Any number of characters

**PART TWO: REFERENCE**

---

**Language*****Enum***

Description

Sets/Queries which scripting language is to be used

Values

JScript

VBScript

**Status*****String***

Range: Any number of characters

Description

Queries status, as reported by the scripting engine (errors).

**Timeout*****Double***

Range: From 1 to 12000, step 0.001

Description

Sets/Queries Timeout for scripting calculation.

**PERCENTILE**     *app.Measure.Px.Operator (ParamEngine = "Percentile")*

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.P1.ParamEngine = "Percentile"

' Set the percentile level to 67%. (i.e. will calculate percentile
  @ 67%)
app.Measure.P1.Operator.HPctPop = 67
```

**HPctPop*****Double***

Range:     From 0 to 100, step 1

## Description

Sets/Queries the percentage of the population that falls to the left of (or below) the desired percentile. For example, the median is the 50<sup>th</sup> percentile, or the horizontal coordinate of the histogram at which 50% of the population falls to the left.

## PART TWO: REFERENCE

**PERIOD@LEVEL***app.Measure.Px.Operator (ParamEngine = "PeriodAtLevel")*

Calculates the period of the input, using a specified level and slope.

<b>AbsLevel</b>	Double
<b>BaseFrequency</b>	Double
<b>FindBaseFrequency</b>	Action
<b>FindLevel</b>	Action
<b>Hysteresis</b>	Double
<b>LevelType</b>	Enum
<b>PercentLevel</b>	Double
<b>SignalType</b>	Enum
<b>Slope</b>	Enum
<b>StdBaseFrequency</b>	Enum
<b>UseBaseFrequency</b>	Enum

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.P1.ParamEngine = "PeriodAtLevel"
```

**AbsLevel****Double**

Range: From -100 to 100, step 1e-007

## Description

Using ParamEngine = "PeriodAtLevel", refer to the corresponding variable for the Delta Time At Level parameter.

**BaseFrequency****Double**

Range: From 1 to 4e+010, step 1000

## Description

Sets/Queries the frequency to be used as a reference when SignalType = "Data"; otherwise, not used.

**FindBaseFrequency****Action**

## Description

Automatically finds (by analyzing the input) the base frequency for a SignalType = "Data".

**FindLevel****Action**

## Description

Causes the engine to find a suitable level for either LevelType ("Absolute" or "Percent")

**Hysteresis****Double**

Range: From 0 to 10, step 0.1

## Description

Sets/Queries the Hysteresis level used for edge or transition detection. (This is an advanced feature.)

**LevelType****Enum**

## Description

Sets/Queries whether the levels of the signal amplitude are "Percent" or "Absolute".

## Values

Absolute

Percent

**PercentLevel****Double**

Range: From 0 to 100, step 1

## Description

Sets/Queries the percent level to be used when LevelType = "Percent".

**SignalType****Enum**

## Description

Using ParamEngine = "PeriodAtLevel", refer to the corresponding variable for the Frequency At Level parameter.

## Values

Clock

Data

**PART TWO: REFERENCE**

---

**Slope****Enum**

## Description

Using ParamEngine = "PeriodAtLevel", refer to the corresponding variable for the Delta Time At Level parameter.

## Values

Neg  
Pos

**StdBaseFrequency****Enum**

## Description

Using ParamEngine = "PeriodAtLevel", refer to the corresponding variable for the Frequency At Level parameter.

## Values

1.544MHz  
139.264MHz  
155.52MHz  
2.048MHz  
2488.32MHz  
34.368MHz  
44.736MHz  
51.84MHz  
622.08MHz  
8.448MHz

**UseBaseFrequency****Enum**

## Description

Sets/Queries which frequency to use, a "Standard" frequency or "Custom" frequency, as specified by BaseFrequency. This control only has effect when the SignalType = "Data".

## Values

Custom  
Standard

**PHASE***app.Measure.Px.Operator (ParamEngine = "Phase")*

<b>OutputType</b>	Enum
<b>RefAbsLevel</b>	Double
<b>RefFindLevel</b>	Action
<b>RefHysteresis</b>	Double
<b>RefLevelType</b>	Enum
<b>RefPercentLevel</b>	Double
<b>RefSlope</b>	Enum
<b>SigAbsLevel</b>	Double
<b>SigFindLevel</b>	Action
<b>SigHysteresis</b>	Double
<b>SigLevelType</b>	Enum
<b>SigPercentLevel</b>	Double
<b>SigSlope</b>	Enum

**OutputType***Enum*

Description

Sets/Queries the output type for Phase Px.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to phase difference.
app.Measure.P1.ParamEngine = "Phase"

' Set the output unit as radians.
app.Measure.P1.Operator.OutputType = "Radians"
```

Values

Degrees  
Percent  
Radians

**RefAbsLevel***Double*

Range: From -100 to 100, step 1e-007

Description

Sets/Queries the test level for the reference trace in absolute units.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to phase difference.
app.Measure.P1.ParamEngine = "Phase"
```

## PART TWO: REFERENCE

---

```
' Set the reference trace test level in absolute units as 10 mV.
app.Measure.P1.Operator.RefAbsLevel = 0.01
```

### RefFindLevel

**Action**

#### Description

Finds the test level for the reference trace.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 to phase difference.
app.Measure.P3.ParamEngine = "Phase"
' Find the test level for the reference trace.
app.Measure.P3.Operator.RefFindLevel
```

### RefHysteresis

**Double**

Range: From 0 to 10, step 0.1

#### Description

Sets/Queries the hysteresis range for the reference trace.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to phase difference.
app.Measure.P1.ParamEngine = "Phase"
' Set the reference hysteresis in graticule divisions.
app.Measure.P1.Operator.RefHysteresis = 0.7
```

### RefLevelType

**Enum**

#### Description

Sets/Queries the unit of measurement for the test level of the reference trace.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to phase difference.
app.Measure.P1.ParamEngine = "Phase"
' Set the reference level to be measured in absolute units.
app.Measure.P1.Operator.RefLevelType = "Absolute"
```

## Values

Absolute  
Percent

**RefPercentLevel****Double**

Range: From 0 to 100, step 1

## Description

Sets/Queries the test level for the reference trace in percent.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 to phase difference.
app.Measure.P3.ParamEngine = "Phase"

' Set the reference test level in percent.
app.Measure.P3.Operator.RefPercentLevel = 55
```

**RefSlope****Enum**

## Description

Sets/Queries the polarity of the measured reference transitions.

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P1 to phase difference.
app.Measure.P1.ParamEngine = "Phase"

' Set the reference slope to negative.
app.Measure.P1.Operator.RefSlope = "Neg"
```

## Values

Both  
Neg  
Pos

**SigAbsLevel****Double**

Range: From -100 to 100, step 1e-007

## Description

Sets/Queries the test level for the signal in absolute units.

**SigFindLevel****Action**

## Description

Causes the engine to find a suitable level for either SigLevelType ("Absolute" or "Percent").

## PART TWO: REFERENCE

---

### SigHysteresis

*Double*

Range: From 0 to 10, step 0.1

Description

Sets/Queries the hysteresis range for the signal.

Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P3 to phase difference.
app.Measure.P3.ParamEngine = "Phase"
' Set the signal hysteresis in graticule divisions.
app.Measure.P3.Operator.SigHysteresis = 0.7
```

### SigLevelType

*Enum*

Description

Sets/Queries which level to use "Percent" or "Absolute" for transitions on the signal.

Values

Absolute  
Percent

### SigPercentLevel

*Double*

Range: From 0 to 100, step 1

Description

Sets/Queries the test level for the signal in percent.

### SigSlope

*Enum*

Description

Sets/Queries the polarity of the measured signal transitions.

Values

Both  
Neg  
Pos

**RISE@LEVEL***app.Measure.Px.Operator (ParamEngine = "RiseAtLevel")*

Calculates the rise time (transition time on a rising edge) using two specified levels.

<b>HighAbs</b>	Double
<b>HighPct</b>	Double
<b>LevelsAre</b>	Enum
<b>LowAbs</b>	Double
<b>LowPct</b>	Double

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
app.Measure.P1.ParamEngine = "RiseAtLevel"
app.Measure.P1.Operator.LevelsAre = "Percent"
app.Measure.P1.Operator.HighPct = 70
app.Measure.P1.Operator.LowPct = 30
```

**HighAbs****Double**

Range: From -100 to 100, step 1e-007

Description

Sets/Queries the higher (second) level crossing used, if LevelsAre = "Absolute".

**HighPct****Double**

Range: From 11 to 95, step 1

Description

Sets/Queries the higher (second) level crossing used, if LevelsAre = "Percent".

**LevelsAre****Enum**

Description

Sets/Queries whether the levels used are "Absolute" or "Percent".

Values

Absolute

Percent

**LowAbs****Double**

Range: From -100 to 100, step 1e-007

Description

Sets/Queries the lower (first) level crossing used, if LevelsAre = "Absolute".

## PART TWO: REFERENCE

---

**LowPct**

**Double**

Range: From 5 to 89, step 1

Description

Sets/Queries the lower (first) level crossing used, if LevelsAre = "Percent".

**RMS** *app.Measure.Px.Operator (ParamEngine = "RootMeanSquare")*

Calculates the root mean square of the values of the input waveform.

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
app.Measure.P1.ParamEngine = "RMS"
```

**Cyclic****Bool****Description**

Sets/Queries whether the RMS is measured over an integral number of cycles (suggested for periodic signals).

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")

' Set parameter P2 to RMS.
app.Measure.P2.ParamEngine = "RMS"

' Set the RMS parameter for cyclic measurements.
app.Measure.P2.ParamEngine.Cyclic = True
```

## PART TWO: REFERENCE

## SETUP

*app.Measure.Px.Operator (ParamEngine = "Setup")*

Calculates the setup time associated with a pair of input waveforms for Clock and Data.

<b>ClockAbsLevel</b>	Double
<b>ClockFindLevel</b>	Action
<b>ClockHysteresis</b>	Double
<b>ClockLevells</b>	Enum
<b>ClockPctLevel</b>	Double
<b>ClockSlope</b>	Enum
<b>CursorDisplay</b>	Enum
<b>DataAbsLevel</b>	Double
<b>DataFindLevel</b>	Action
<b>DataHysteresis</b>	Double
<b>DataLevells</b>	Enum
<b>DataPctLevel</b>	Double
<b>DataSlope</b>	Enum
<b>Summary</b>	String

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
app.Measure.Pl.ParamEngine = "Setup"
```

**ClockAbsLevel****Double**

Range: From -100 to 100, step 1e-007

Description

Sets/Queries the level used if ClockLevells = "Absolute".

**ClockFindLevel****Action**

Description

Causes the engine to find a suitable level for either ClockLevells ("Absolute" or "Percent").

**ClockHysteresis****Double**

Range: From 0 to 10, step 0.1

Description

Using ParamEngine = "Setup", refer to the corresponding variable for the Hold Time parameter.

**ClockLevell****Enum**

## Description

Sets/Queries whether the Clock signal levels are specified in "Percent" or "Absolute".

## Values

Absolute  
Percent

**ClockPctLevel****Double**

Range: From 0 to 100 step 1

## Description

Using ParamEngine = "Setup", please refer to the corresponding variable for the Hold Time parameter.

**ClockSlope****Enum**

## Description

Sets/Queries the polarity of transitions of the Clock signal.

## Values

Both  
Neg  
Pos

**CursorDisplay****Enum**

## Description

Using ParamEngine = "Setup", refer to the corresponding variable for the Hold Time parameter.

## Values

Detailed  
Off  
Simple

**DataAbsLevel****Double**

Range: From -100 to 100, step 1e-007

## Description

Sets/Queries the level used if the ClockLevell = "Percent"

**PART TWO: REFERENCE**

---

**DataFindLevel****Action**

Description

Causes the engine to find a suitable level for either DataLevels ("Absolute" or "Percent")

**DataHysteresis****Double**

Range: From 0 to 10, step 0.1

Description

Using ParamEngine = "Setup", refer to the corresponding variable for the Hold Time parameter.

**DataLevels****Enum**

Description

Sets/Queries whether the Data signal level is DataAbsLevel or DataPctLevel.

Values

Absolute

Percent

**DataPctLevel****Double**

Range: From 0 to 100, step 1

Description

Using ParamEngine = "Setup", refer to the corresponding variable for the Hold Time parameter.

**DataSlope****Enum**

Description

Sets/Queries the polarity of transitions to be used for the Data signal.

Values

Both

Neg

Pos

**Summary****String**

Range: Any number of characters

Description

Using ParamEngine = "Setup", refer to the corresponding variable for the Hold Time parameter.

**SKEW***app.Measure.Px.Operator (ParamEngine = "Skew")*

Calculates the skew between two clock signal waveforms.

<b>Clock1AbsLevel</b>	Double
<b>Clock1FindLevel</b>	Action
<b>Clock1Hysteresis</b>	Double
<b>Clock1Levells</b>	Enum
<b>Clock1PctLevel</b>	Double
<b>Clock1Slope</b>	Enum
<b>Clock2AbsLevel</b>	Double
<b>Clock2FindLevel</b>	Action
<b>Clock2Hysteresis</b>	Double
<b>Clock2Levells</b>	Enum
<b>Clock2PctLevel</b>	Double
<b>Clock2Slope</b>	Enum

## Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
app.Measure.P1.ParamEngine = "Skew"
```

**Clock1AbsLevel****Double**

Range: From -100 to 100, step 1e-007

**Clock1FindLevel****Action**

## Description

Automatically find a suitable level for Clock1, for either "Percent" or "Absolute" levels.

**Clock1Hysteresis****Double**

Range: From 0 to 10, step 0.1

## Description

Sets/Queries hysteresis for transition detection used for Clock1.

**Clock1Levells****Enum**

## Description

Sets/Queries whether to use Percent or Absolute levels for Clock1.

## Values

Absolute  
Percent

**PART TWO: REFERENCE**

---

**Clock1PctLevel****Double**

Range: From 0 to 100, step 1

Description

Sets/Queries the "Percent" of the amplitude of Clock1 to use for a transition level, if Clock1Levells = "Percent".

**Clock1Slope****Enum**

Description

Sets/Queries the polarity of transitions detected on Clock1.

Values

Both

Neg

Pos

**Clock2AbsLevel****Double**

Range: From -100 to 100, step 1e-007

Description

Using ParamEngine = "Skew", refer to the corresponding variable for the Hold Time parameter.

**Clock2FindLevel****Action**

Description

Using ParamEngine = "Skew", refer to the corresponding variable for the Hold Time parameter.

**Clock2Hysteresis****Double**

Range: From 0 to 10, step 0.1

Description

Sets/Queries hysteresis for transition detection used for Clock1.

**Clock2Levells****Enum**

Description

Sets/Queries whether to use Percent or Absolute levels for Clock2.

Values

Absolute

Percent

**Clock2PctLevel****Double**

Range: From 0 to 100, step 1

Description

Sets/Queries the "Percent" of the amplitude of Clock2 to use for a transition level, if Clock2Levells = "Percent".

**Clock2Slope***Enum*

## Description

Sets/Queries the polarity of transitions detected on Clock2.

## Values

Both

Neg

Pos

## PART TWO: REFERENCE

---

### **STD DEV**    *app.Measure.Px.Operator (ParamEngine = "StandardDeviation")*

Calculates the standard deviation of the input waveform's vertical values.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
' Set parameter P1 to standard deviation.
app.Measure.P1.ParamEngine = "StandardDeviation"
' Set the RMS parameter for cyclic measurements.
app.Measure.P1.Operator.Cyclic = True
```

#### **Cyclic**

**Bool**

#### Description

Sets/Queries whether the standard deviation is to be measured over an integral number of complete cycles.

**TIE@LEVEL***app.Measure.Px.Operator (ParamEngine = "TIE")*

Calculates the Time Interval Error: the errors in observed transition times relative to a series of expected times (virtual clock).

<b>AbsLevel</b>	Double
<b>BaseFrequency</b>	Double
<b>CutOffDivisor</b>	Double
<b>DatalsNRZ</b>	Bool
<b>FindBaseFrequency</b>	Action
<b>FindLevel</b>	Action
<b>Hysteresis</b>	Double
<b>IncludeVirtualEdges</b>	Bool
<b>LevelType</b>	Enum
<b>PercentLevel</b>	Double
<b>ResultScaling</b>	Enum
<b>SignalType</b>	Enum
<b>Slope</b>	Enum
<b>StdBaseFrequency</b>	Enum
<b>UseBaseFrequency</b>	Enum
<b>UseGoldenPLL</b>	Bool

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
app.Measure.Pl.ParamEngine = "TIE"
```

**AbsLevel****Double**

Range: From -100 to 100, step 1e-007

## Description

Sets/Queries the absolute transition level to be used when LevelType = "Absolute".

**BaseFrequency****Double**

Range: From 1 to 4e+010, step 1000

## Description

Using ParamEngine = "TIE", refer to the corresponding variable for the Delta Period At Level parameter.

**CutOffDivisor****Double**

Range: From 20 to 10000, step 1

## Description

Sets/Queries the cut-off divisor for the "Golden PLL".

**PART TWO: REFERENCE**

---

**DatalsNRZ*****Bool***

## Description

Sets/Queries whether the signal is of NRZ type.

**FindBaseFrequency*****Action***

## Description

Automatically determines the Base Frequency for the expected virtual clock.

**FindLevel*****Action***

## Description

Using ParamEngine = "TIE", refer to the corresponding variable for the Delta Period At Level parameter.

**Hysteresis*****Double***

Range: From 0 to 10, step 0.1

## Description

Sets/Queries hysteresis for transition detection used for the input waveform.

**IncludeVirtualEdges*****Bool***

## Description

Sets/Queries whether to include virtual edges in the series of interval errors reported by this measurement.

**LevelType*****Enum***

## Description

Sets/Queries whether to use Percent or Absolute levels.

## Values

Absolute

Percent

**PercentLevel*****Double***

Range: From 0 to 100, step 1

## Description

Sets/Queries the "Percent" of the amplitude of the input waveform for a transition detection, if LevelType = "Percent".

**ResultScaling****Enum**

## Description

Sets/Queries whether the measurement is to be displayed in seconds (Time) or in unit intervals (UI).

## Values

Time  
UI

**SignalType****Enum**

## Description

Sets/Queries whether the input signal is a Clock or Data.

## Values

Clock  
Data

**Slope****Enum**

## Description

Sets/Queries the polarity of transitions detected.

## Values

Both  
Neg  
Pos

**StdBaseFrequency****Enum**

## Description

Sets/Queries a standard base frequency from a list of frequencies. This value is only used if the UseBaseFrequency = "Standard".

## Values

1.544MHz  
139.264MHz  
155.52MHz  
2.048MHz  
2488.32MHz  
34.368MHz  
44.736MHz  
51.84MHz  
622.08MHz  
8.448MHz

**PART TWO: REFERENCE**

---

**UseBaseFrequency*****Enum***

## Description

Sets/Queries whether to use the custom base frequency or one chosen from a list of standard frequencies.

## Values

Custom  
Standard

**UseGoldenPLL*****Bool***

## Description

Sets/Queries whether to use a golden PLL for the expected times.

**TIME@LEVEL***app.Measure.Px.Operator (ParamEngine = "TimeAtLevel")*

<b>AbsLevel</b>	Double
<b>FindLevel</b>	Action
<b>Hysteresis</b>	Double
<b>HysteresisPct</b>	Double
<b>HysteresisType</b>	Enum
<b>LevelType</b>	Enum
<b>PercentLevel</b>	Double
<b>Slope</b>	Enum

**AbsLevel****Double**

Range: From -100 to 100, step 1e-007

Description

Sets/Queries the level for transitions detected on the input waveform if LevelType = "Absolute".

**FindLevel****Action**

Description

Using ParamEngine = "TimeAtLevel", please refer to the corresponding variable for the Delta Time At Level parameter.

**Hysteresis****Double**

Range: From 0 to 10, step 0.1

Description

Sets/Queries hysteresis for transition detection used.

**HysteresisPct****Double**

Range: From 0 to 100, step 0.1

Description

Sets/Queries the hysteresis as a percentage of signal amplitude, if HysteresisType = "Percent".

**HysteresisType****Enum**

Description

Sets/Queries whether hysteresis is specified in divisions, or if it is a percentage of the input waveform amplitude.

Values

Divisions

Percent

**PART TWO: REFERENCE**

---

**LevelType*****Enum***

## Description

Using ParamEngine = "TimeAtLevel", refer to the corresponding variable for the Delta Time At Level parameter.

## Values

Absolute  
Percent

**PercentLevel*****Double***

Range: From 0 to 100, step 1

## Description

Sets/Queries the "Percent" of the amplitude of the input waveform is used for detecting transitions if LevelType = "Percent".

**Slope*****Enum***

## Description

Sets/Queries the polarity of transitions detected.

## Values

Both  
Neg  
Pos

**WIDTH@LEVEL**     *app.Measure.Px.Operator (ParamEngine = "WidthAtLevel")*

Calculates the width of pulses in the input waveform at specified levels.

<b>AbsLevel</b>	Double
<b>FindLevel</b>	Action
<b>FirstWhenBoth</b>	Enum
<b>Hysteresis</b>	Double
<b>HysteresisPct</b>	Double
<b>HysteresisType</b>	Enum
<b>LevelType</b>	Enum
<b>PercentLevel</b>	Double
<b>Slope</b>	Enum

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
' Set parameter P1 to width at level.
app.Measure.P1.ParamEngine = "WidthAtLevel"
' Set first used polarity to negative.
app.Measure.P1.Operator.Slope = "Both"
app.Measure.P1.Operator.FirstWhenBoth = "Neg"
```

**AbsLevel****Double**

Range:     From -100 to 100 step 1e-007

## Description

Sets/Queries the level for transitions detected on the input waveform, if LevelType = "Absolute".

**FindLevel****Action**

## Description

Automatically finds a suitable level for the width measurement, for either case of LevelType.

**FirstWhenBoth****Enum**

## Description

Sets/Queries the polarity of the first accepted transition when both polarities are used.

## Values

Neg  
Pos

**PART TWO: REFERENCE**

---

**Hysteresis****Double**

Range: From 0 to 10, step 0.1

## Description

Sets/Queries hysteresis for transition detection if HysteresisType = "Absolute".

**HysteresisPct****Double**

Range: From 0 to 100, step 0.1

## Description

Sets/Queries hysteresis in percent for transition detection if HysteresisType = "Percent".

**HysteresisType****Enum**

## Description

Sets/Queries whether to use Percent or Divisions for hysteresis.

## Values

Divisions

Percent

**LevelType****Enum**

## Description

Sets/Queries whether to use Percent or Absolute levels.

## Values

Absolute

Percent

**PercentLevel****Double**

Range: From 0 to 100, step 1

## Description

Sets/Queries the "Percent" of the amplitude of the input waveform, if LevelType = "Percent".

**Slope****Enum**

## Description

Sets/Queries the polarity of transitions detected.

## Values

Both

Neg

Pos

**X AT PEAK***app.Measure.Px.Operator (ParamEngine = "XAtPeak")*

Estimates the horizontal position of the N<sup>th</sup> most significant peak in a distribution

**Example**

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.MeasureMode = "MyMeasure"
app.Measure.Pl.ParamEngine = "XAtPeak"
' Set the parameter to measure location of the 4th peak.
app.Measure.Pl.Operator.PeakNumber = 4
```

**PeakNumber***Integer*

Range: From 1 to 10000, step 1

**Description**

Sets/Queries the horizontal position of the N<sup>th</sup> highest peak of a histogram.

## PART TWO: REFERENCE

---

### ZERO LEVEL *app.Measure.Px.Operator (ParamEngine = "EyeZeroLevel")*

Estimates the low level for the (assumed input) eye diagram, based on a slice through the middle of the eye diagram of specified width, PctCutWidth.

#### Example

```
' Visual Basic Script
Set app = CreateObject("LeCroy.XStreamDSO")
app.Measure.Pl.ParamEngine = "EyeZeroLevel"
app.Measure.Pl.Operator.PctCutWidth = 8
```

#### PctCutWidth

*Double*

Range: From 0 to 100, step 0.1

#### Description

Sets/Queries the width of the vertical slice through the eye, from which the zero level is estimated.



## APPENDIX A

## Glossary

**ACTIVEX**

Microsoft's brand name for the technologies that enable interoperability using the Component Object Model (COM). ActiveX technology includes, but is not limited to, OLE.

**AUTOMATION**

COM-based technology that enables binding at run time, or late binding, to an object's methods and properties and also makes possible cross-application macro programming. Formerly referred to as OLE Automation.

**AUTOMATION CLIENT**

An application, programming tool, or scripting language that accesses services provided by Automation objects. Formerly referred to as Automation controller.

**AUTOMATION OBJECT**

An instance of a class defined within an application that is exposed for access by other applications or programming tools by Automation interfaces.

**AUTOMATION SERVER**

An application, type library, or other source that makes Automation objects available for programming by other applications, programming tools, or scripting languages.

**COM (COMPONENT OBJECT MODEL)**

The programming model and binary standard on which OLE is based. COM defines how objects and their clients interact within processes or across process boundaries.

**DCOM (DISTRIBUTED COMPONENT OBJECT MODEL)**

Distributed form of COM, enables communication between computers using the COM standard.

**DISPATCH INTERFACES (DISPINTERFACE)**

An IDispatch interface that responds only to a certain fixed set of names. The properties and methods of the dispinterface are not in the virtual function table (VTBL) for the object.

**EARLY BINDING**

Also known as static binding (converting symbolic addresses in the program to storage-related addresses) this is binding that occurs during program compilation or linkage.

**LATE BINDING**

The ability to bind member names to dispatch identifiers (IDs) at run time, rather than at compile time.

