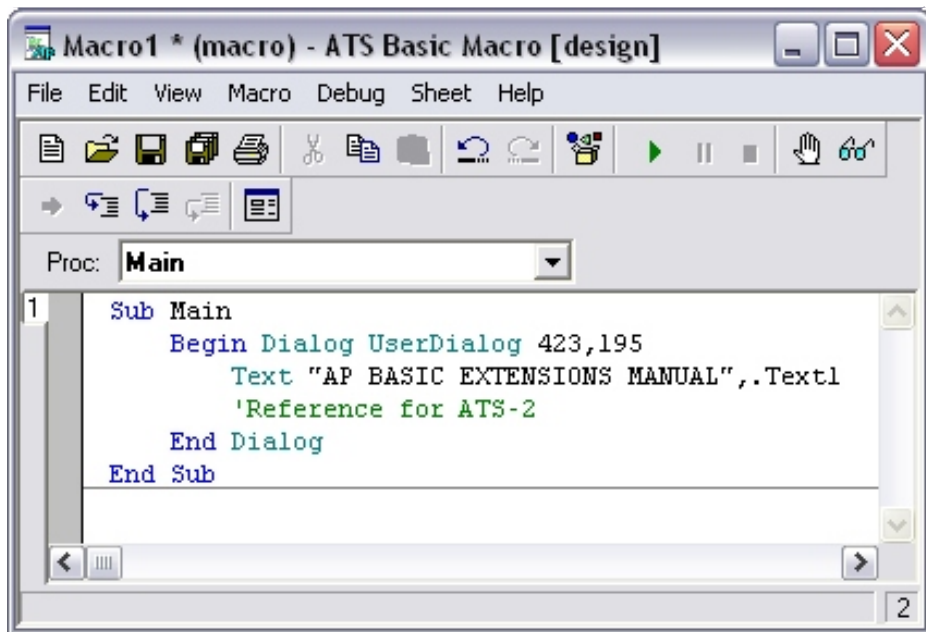


# ATS-2

## AP Basic Extensions Manual for the ATS-2



The screenshot shows a window titled "Macro1 \* (macro) - ATS Basic Macro [design]". The menu bar includes File, Edit, View, Macro, Debug, Sheet, and Help. The toolbar contains icons for file operations (New, Open, Save, Print, Copy, Paste, Undo, Redo) and execution (Run, Stop, Break). A dropdown menu shows "Proc: Main". The code editor contains the following text:

```
1 Sub Main
  Begin Dialog UserDialog 423,195
    Text "AP BASIC EXTENSIONS MANUAL",.Text1
    'Reference for ATS-2
  End Dialog
End Sub
```

The window has a status bar at the bottom right showing the number "2".



# AP Basic Extensions Reference for ATS-2



Copyright © 2001-2004 Audio Precision, Inc.

All rights reserved.

Document part number 8211.0137 Revision 2

ATS Versions 1.2 and 1.4

No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the publisher.

Audio Precision®, System One®, System Two™, System Two Cascade™, System One + DSP™, System Two + DSP™, Dual Domain®, FASTTEST®, and APWIN™ are trademarks of Audio Precision, Inc. Windows is a trademark of Microsoft Corporation.

Published by:



Audio Precision, Inc.  
5750 SW Arctic Drive  
Beaverton, Oregon 97005  
U.S. Toll Free: 1-800-231-7350  
Tel: (503) 627-0832 Fax: (503) 641-8906  
email: [info@audioprecision.com](mailto:info@audioprecision.com)  
web: [audioprecision.com](http://audioprecision.com)

# Contents

## Chapter 1

### ATS Extensions Reference

Introduction . . . . .	1-1
Manual Conventions . . . . .	1-2

## Chapter 2

System Panels . . . . .	2-1
-------------------------	-----

## Chapter 3

### Application

ATS.Application.AppDir . . . . .	3-1
ATS.Application.ClearCurrentError . . . . .	3-2
ATS.Application.CloseAll . . . . .	3-2
ATS.Application.CloseAllPages . . . . .	3-3
ATS.Application.CopyPanelToClipboard . . . . .	3-4
ATS.Application.DisplayCurrentError . . . . .	3-4
ATS.Application.DisplayDataOnTestOpen . . . . .	3-5
ATS.Application.DoReadings . . . . .	3-6
ATS.Application.GetCurrentErrorString . . . . .	3-8
ATS.Application.GetEnumString . . . . .	3-8
ATS.Application.Input . . . . .	3-9
ATS.Application.MacroDir . . . . .	3-10
ATS.Application.Name . . . . .	3-11
ATS.Application.NewData . . . . .	3-12
ATS.Application.NewMacro . . . . .	3-12
ATS.Application.NewTest . . . . .	3-13
ATS.Application.Output . . . . .	3-13
ATS.Application.Page . . . . .	3-14
ATS.Application.PanelClose . . . . .	3-15
ATS.Application.PanelOpen . . . . .	3-16
ATS.Application.Quit . . . . .	3-16
ATS.Application.Restore . . . . .	3-17
ATS.Application.SetWatchDogTimer . . . . .	3-18

ATS.Application.SysType . . . . .	3-19
ATS.Application.TestDir . . . . .	3-19
ATS.Application.TestName . . . . .	3-20
ATS.Application.ThrowErrors . . . . .	3-20
ATS.Application.Version . . . . .	3-22
ATS.Application.Visible . . . . .	3-23
ATS.Application.VisibleMacroEditor . . . . .	3-23
ATS.Application.VisibleOnTestLoad . . . . .	3-24
ATS.Application.WorkingDir . . . . .	3-25

## Chapter 4

### Auxiliary Instrument

ATS.Aux.ReadingRdg . . . . .	4-1
ATS.Aux.ReadingReady . . . . .	4-3
ATS.Aux.ReadingSettling . . . . .	4-4
ATS.Aux.ReadingTrig . . . . .	4-4
ATS.Aux.SetReading . . . . .	4-5
ATS.Aux.Setting . . . . .	4-5

## Chapter 5

### Bar Graph

ATS.BarGraph.AxisAutoScale . . . . .	5-1
ATS.BarGraph.AxisIncrement . . . . .	5-3
ATS.BarGraph.AxisLeft . . . . .	5-3
ATS.BarGraph.AxisLogLin . . . . .	5-4
ATS.BarGraph.AxisRight . . . . .	5-4
ATS.BarGraph.Comment . . . . .	5-5
ATS.BarGraph.CommentShow . . . . .	5-6
ATS.BarGraph.DigitsOnly . . . . .	5-7
ATS.BarGraph.Id . . . . .	5-7
ATS.BarGraph.Max . . . . .	5-7
ATS.BarGraph.Min . . . . .	5-8
ATS.BarGraph.New . . . . .	5-8
ATS.BarGraph.Reset . . . . .	5-9
ATS.BarGraph.TargetLower . . . . .	5-9
ATS.BarGraph.TargetRange . . . . .	5-10
ATS.BarGraph.TargetUpper . . . . .	5-10
ATS.BarGraph.Title . . . . .	5-11

## Chapter 6

### RS-232 Communication

ATS.Comm.Break . . . . .	6-1
ATS.Comm.CD Holding . . . . .	6-1
ATS.Comm.CDTimeout . . . . .	6-2
ATS.Comm.CommError . . . . .	6-2
ATS.Comm.CommId . . . . .	6-3
ATS.Comm.CommPort . . . . .	6-4

ATS.Comm.CTSHolding	6-5
ATS.Comm.CTSTimeout	6-5
ATS.Comm.DSRHolding	6-6
ATS.Comm.DSRTimeout	6-6
ATS.Comm.DTREnable	6-6
ATS.Comm.Handshaking	6-7
ATS.Comm.InBufferCount	6-8
ATS.Comm.InBufferSize	6-8
ATS.Comm.Input	6-9
ATS.Comm.InputLen	6-9
ATS.Comm.Interval	6-10
ATS.Comm.NullDiscard	6-10
ATS.Comm.OutBufferCount	6-10
ATS.Comm.OutBufferSize	6-11
ATS.Comm.Output	6-11
ATS.Comm.ParityReplace	6-12
ATS.Comm.PortOpen	6-12
ATS.Comm.RTSEnable	6-13
ATS.Comm.Settings	6-13

## Chapter 7

### Computes

ATS.Compute.Avg.Apply	7-1
ATS.Compute.Avg.Data	7-2
ATS.Compute.Avg.PostSweep	7-2
ATS.Compute.Avg.Start	7-3
ATS.Compute.Avg.StartUnit	7-3
ATS.Compute.Avg.Stop	7-4
ATS.Compute.Avg.StopUnit	7-4
ATS.Compute.Center.Apply	7-5
ATS.Compute.Center.Data	7-5
ATS.Compute.Center.PostSweep	7-6
ATS.Compute.Center.Start	7-6
ATS.Compute.Center.StartUnit	7-7
ATS.Compute.Center.Stop	7-7
ATS.Compute.Center.StopUnit	7-8
ATS.Compute.Clear.All	7-8
ATS.Compute.Delta.Apply	7-9
ATS.Compute.Delta.Data	7-9
ATS.Compute.Delta.DataColumn	7-10
ATS.Compute.Delta.FileName	7-10
ATS.Compute.Delta.PostSweep	7-11
ATS.Compute.Equalize.Apply	7-11
ATS.Compute.Equalize.Data	7-12
ATS.Compute.Equalize.DataColumn	7-13
ATS.Compute.Equalize.FileName	7-13
ATS.Compute.Equalize.PostSweep	7-14

ATS.Compute.Invert.Apply . . . . .	7-14
ATS.Compute.Invert.Data . . . . .	7-15
ATS.Compute.Invert.Horizontal . . . . .	7-16
ATS.Compute.Invert.HorizontalUnit . . . . .	7-16
ATS.Compute.Invert.PostSweep . . . . .	7-16
ATS.Compute.Linearity.Apply . . . . .	7-17
ATS.Compute.Linearity.Data . . . . .	7-18
ATS.Compute.Linearity.PostSweep . . . . .	7-18
ATS.Compute.Linearity.Start . . . . .	7-19
ATS.Compute.Linearity.StartUnit . . . . .	7-19
ATS.Compute.Linearity.Stop . . . . .	7-20
ATS.Compute.Linearity.StopUnit . . . . .	7-20
ATS.Compute.Max.Apply . . . . .	7-21
ATS.Compute.Max.Data . . . . .	7-21
ATS.Compute.Max.PostSweep . . . . .	7-22
ATS.Compute.Max.Start . . . . .	7-23
ATS.Compute.Max.StartUnit . . . . .	7-23
ATS.Compute.Max.Stop . . . . .	7-23
ATS.Compute.Max.StopUnit . . . . .	7-24
ATS.Compute.Min.Apply . . . . .	7-24
ATS.Compute.Min.Data . . . . .	7-25
ATS.Compute.Min.PostSweep . . . . .	7-26
ATS.Compute.Min.Start . . . . .	7-26
ATS.Compute.Min.StartUnit . . . . .	7-27
ATS.Compute.Min.Stop . . . . .	7-27
ATS.Compute.Min.StopUnit . . . . .	7-27
ATS.Compute.Normalize.Apply . . . . .	7-28
ATS.Compute.Normalize.Data . . . . .	7-29
ATS.Compute.Normalize.Horizontal . . . . .	7-29
ATS.Compute.Normalize.HorizontalUnit . . . . .	7-30
ATS.Compute.Normalize.PostSweep . . . . .	7-30
ATS.Compute.Normalize.Target . . . . .	7-31
ATS.Compute.Normalize.TargetUnit . . . . .	7-31
ATS.Compute.Smooth.Apply . . . . .	7-32
ATS.Compute.Smooth.Auto . . . . .	7-32
ATS.Compute.Smooth.Data . . . . .	7-33
ATS.Compute.Smooth.Passes . . . . .	7-33
ATS.Compute.Smooth.PostSweep . . . . .	7-34
ATS.Compute.Status.Id . . . . .	7-34
ATS.Compute.Status.NumOf . . . . .	7-36

## Chapter 8

### Data

ATS.Data.AddRowToEnd . . . . .	8-1
ATS.Data.ColLimitError . . . . .	8-2
ATS.Data.ColLowerLimitError . . . . .	8-3
ATS.Data.ColName . . . . .	8-3



ATS.Data.ColNumOf	8-4
ATS.Data.ColSize	8-4
ATS.Data.ColUnit	8-5
ATS.Data.ColUpperLimitError	8-6
ATS.Data.DeleteRow	8-6
ATS.Data.Id	8-8
ATS.Data.InsertRowAfter	8-9
ATS.Data.InsertRowBefore	8-9
ATS.Data.LimitError	8-10
ATS.Data.LowerLimitError	8-11
ATS.Data.Status	8-11
ATS.Data.Transfer	8-12
ATS.Data.UpdateDisplay	8-14
ATS.Data.UpperLimitError	8-15
ATS.Data.Value	8-16

## Chapter 9

### Events

ATSEvent_OnAuxSetting	9-1
ATSEvent_OnDcxProgramControllInput	9-2
ATSEvent_OnError	9-3
ATSEvent_OnSweepEnd	9-4
ATSEvent_OnSweepNestEnd	9-5
ATSEvent_OnSweepNestStart	9-5
ATSEvent_OnSweepStart	9-6
ATSEvent_OnSweepStep	9-6
ATSEvent_OnSweepStepEnd	9-6
ATSEvent_OnSweepTrigger	9-6
ATSEvent_OnWatchDogTimeout	9-7

## Chapter 10

### File

ATS.File.AppendData	10-1
ATS.File.ExportASCIIData	10-1
ATS.File.ExportGraphic	10-2
ATS.File.ImportASCIIData	10-3
ATS.File.ImportTest	10-4
ATS.File.OpenData	10-4
ATS.File.OpenMacro	10-5
ATS.File.OpenTest	10-6
ATS.File.OpenWfm	10-7
ATS.File.SaveAll	10-8
ATS.File.SaveDataAs	10-9
ATS.File.SaveTest	10-9
ATS.File.SaveTestAs	10-10
ATS.File.SaveWfmAs	10-11

---

## Chapter 11

### Graph

ATS.Graph.Comment . . . . .	11-1
ATS.Graph.CommentAppend . . . . .	11-2
ATS.Graph.CommentShow . . . . .	11-3
ATS.Graph.CopyToSweepPanel . . . . .	11-3
ATS.Graph.CursorPosition . . . . .	11-4
ATS.Graph.CursorRow . . . . .	11-5
ATS.Graph.CursorsOn . . . . .	11-5
ATS.Graph.CursorValue . . . . .	11-6
ATS.Graph.Label . . . . .	11-6
ATS.Graph.LabelAuto . . . . .	11-7
ATS.Graph.Legend.Comment . . . . .	11-8
ATS.Graph.Legend.LineColor . . . . .	11-8
ATS.Graph.Legend.LineStyle . . . . .	11-9
ATS.Graph.Legend.LineThickness . . . . .	11-9
ATS.Graph.OptimizeIndividually . . . . .	11-10
ATS.Graph.OptimizeLeft . . . . .	11-10
ATS.Graph.OptimizeRight . . . . .	11-10
ATS.Graph.OptimizeTogether . . . . .	11-11
ATS.Graph.RefDataClear . . . . .	11-11
ATS.Graph.RefDataShow . . . . .	11-12
ATS.Graph.RefDataStore . . . . .	11-12
ATS.Graph.ScrollBarsOn . . . . .	11-12
ATS.Graph.Sweeps . . . . .	11-13
ATS.Graph.SweepShow . . . . .	11-13
ATS.Graph.SweepTraces . . . . .	11-13
ATS.Graph.TraceShow . . . . .	11-14
ATS.Graph.ZoomOriginal . . . . .	11-14
ATS.Graph.ZoomOut . . . . .	11-14

## Chapter 12

### Log

ATS.LogFile.AddEntry . . . . .	12-1
ATS.LogFile.AddEntryWithoutTimeDate . . . . .	12-2
ATS.LogFile.Clear . . . . .	12-2
ATS.LogFile.Data . . . . .	12-2
ATS.LogFile.Enable . . . . .	12-3
ATS.LogFile.ErrorMessage . . . . .	12-3
ATS.LogFile.FileActivity . . . . .	12-3
ATS.LogFile.FileName . . . . .	12-4
ATS.LogFile.GraphTitle . . . . .	12-4
ATS.LogFile.PassFailMessages . . . . .	12-4
ATS.LogFile.PrintLogFile . . . . .	12-5
ATS.LogFile.TestName . . . . .	12-6
ATS.LogFile.View . . . . .	12-6

---

## Chapter 13

### Macro

ATS.Macro.IsRunning . . . . .	13-1
ATS.Macro.Name . . . . .	13-2

## Chapter 14

### Print

ATS.Printout.Data . . . . .	14-1
ATS.Printout.Graph . . . . .	14-1
ATS.Printout.LoadFromTest . . . . .	14-2
ATS.Printout.TrackGraph . . . . .	14-2

## Chapter 15

### Prompt

ATS.Prompt.FontSize . . . . .	15-1
ATS.Prompt.Hide . . . . .	15-1
ATS.Prompt.IsUp . . . . .	15-2
ATS.Prompt.Position . . . . .	15-2
ATS.Prompt.Show . . . . .	15-3
ATS.Prompt.ShowWithContinue . . . . .	15-3
ATS.Prompt.ShowWithContinueAndStopSweep . . . . .	15-3
ATS.Prompt.Text . . . . .	15-4
ATS.Prompt.Title . . . . .	15-4

## Chapter 16

### Regulation

ATS.Reg.IsRunning . . . . .	16-1
ATS.Reg.SourceHigh . . . . .	16-2
ATS.Reg.SourceId . . . . .	16-2
ATS.Reg.SourceIteration . . . . .	16-3
ATS.Reg.SourceLow . . . . .	16-3
ATS.Reg.SourceOperation . . . . .	16-4
ATS.Reg.SourceStepSize . . . . .	16-5
ATS.Reg.Start . . . . .	16-6
ATS.Reg.StartNoWait . . . . .	16-6
ATS.Reg.SweepEnable . . . . .	16-6
ATS.Reg.TargetId . . . . .	16-7
ATS.Reg.TargetTolerance . . . . .	16-7
ATS.Reg.TargetToleranceMode . . . . .	16-8
ATS.Reg.TargetValue . . . . .	16-8
ATS.Reg.TimeOut . . . . .	16-9

## Chapter 17

### Sweep

ATS.Sweep.AbortTime . . . . .	17-1
ATS.Sweep.Append . . . . .	17-2
ATS.Sweep.CopySettings . . . . .	17-3

---

ATS.Sweep.CreateGraph . . . . .	17-4
ATS.Sweep.CreateTable . . . . .	17-5
ATS.Sweep.Data.AutoDiv . . . . .	17-5
ATS.Sweep.Data.Autoscale . . . . .	17-6
ATS.Sweep.Data.Bottom . . . . .	17-6
ATS.Sweep.Data.Div . . . . .	17-7
ATS.Sweep.Data.Id . . . . .	17-7
ATS.Sweep.Data.Limit.Column . . . . .	17-7
ATS.Sweep.Data.Limit.FileName . . . . .	17-8
ATS.Sweep.Data.Limits . . . . .	17-8
ATS.Sweep.Data.LogLin . . . . .	17-9
ATS.Sweep.Data.Top . . . . .	17-10
ATS.Sweep.External.StartOnRule . . . . .	17-11
ATS.Sweep.GraphType . . . . .	17-13
ATS.Sweep.IsRunning . . . . .	17-13
ATS.Sweep.PreSweepDelay . . . . .	17-13
ATS.Sweep.Recompare . . . . .	17-14
ATS.Sweep.Repeat . . . . .	17-14
ATS.Sweep.Reprocess . . . . .	17-16
ATS.Sweep.Retransform . . . . .	17-17
ATS.Sweep.SinglePoint . . . . .	17-18
ATS.Sweep.Source.AutoDiv . . . . .	17-19
ATS.Sweep.Source.Div . . . . .	17-19
ATS.Sweep.Source.EndOn . . . . .	17-20
ATS.Sweep.Source.Id . . . . .	17-21
ATS.Sweep.Source.LogLin . . . . .	17-21
ATS.Sweep.Source.MinLevel . . . . .	17-22
ATS.Sweep.Source.MinLevelId . . . . .	17-23
ATS.Sweep.Source.Multiply . . . . .	17-23
ATS.Sweep.Source.Spacing . . . . .	17-24
ATS.Sweep.Source.Start . . . . .	17-24
ATS.Sweep.Source.Steps . . . . .	17-25
ATS.Sweep.Source.StepSize . . . . .	17-25
ATS.Sweep.Source.Stop . . . . .	17-26
ATS.Sweep.Source.Table . . . . .	17-26
ATS.Sweep.Source.TableColumn . . . . .	17-27
ATS.Sweep.Source.TableFileName . . . . .	17-27
ATS.Sweep.Spectrum . . . . .	17-28
ATS.Sweep.Start . . . . .	17-29
ATS.Sweep.StartNoWait . . . . .	17-30
ATS.Sweep.StartWithAppend . . . . .	17-30
ATS.Sweep.StartWithRepeat . . . . .	17-31
ATS.Sweep.Stereo . . . . .	17-31
ATS.Sweep.Stop . . . . .	17-31
ATS.Sweep.Timeout . . . . .	17-32
ATS.Sweep.Waveform . . . . .	17-33

---

## Chapter 18

### Analog Generator

ATS2.AGen.Ampl . . . . .	18-1
ATS2.AGen.BurstInterval . . . . .	18-2
ATS2.AGen.BurstLevel . . . . .	18-3
ATS2.AGen.BurstOnTime . . . . .	18-3
ATS2.AGen.ChBTrackA . . . . .	18-4
ATS2.AGen.Config . . . . .	18-5
ATS2.AGen.DACBandwidth . . . . .	18-6
ATS2.AGen.DACSampleRate . . . . .	18-7
ATS2.AGen.DualAmplRatio . . . . .	18-8
ATS2.AGen.EqAmpl . . . . .	18-8
ATS2.AGen.EqCurve . . . . .	18-9
ATS2.AGen.EqCurveColumn . . . . .	18-9
ATS2.AGen.EqCurveFilename . . . . .	18-10
ATS2.AGen.Freq . . . . .	18-10
ATS2.AGen.IMFreq . . . . .	18-11
ATS2.AGen.IMHighFreq . . . . .	18-12
ATS2.AGen.Impedance . . . . .	18-13
ATS2.AGen.Output . . . . .	18-13
ATS2.AGen.OutputOn . . . . .	18-14
ATS2.AGen.Phase . . . . .	18-14
ATS2.AGen.RefdBm . . . . .	18-15
ATS2.AGen.RefdBr . . . . .	18-16
ATS2.AGen.RefdBrAuto . . . . .	18-17
ATS2.AGen.RefFreq . . . . .	18-18
ATS2.AGen.RefFreqAuto . . . . .	18-19
ATS2.AGen.RefWatts . . . . .	18-20
ATS2.AGen.Wfm . . . . .	18-21
ATS2.AGen.WfmName . . . . .	18-22

## Chapter 19

### Analog Input

ATS2.AnalogIn.DCCoupled . . . . .	19-1
ATS2.AnalogIn.Impedance . . . . .	19-2
ATS2.AnalogIn.PeakRdg . . . . .	19-2
ATS2.AnalogIn.PeakReady . . . . .	19-3
ATS2.AnalogIn.PeakTrig . . . . .	19-4
ATS2.AnalogIn.Range . . . . .	19-4
ATS2.AnalogIn.RangeAuto . . . . .	19-4
ATS2.AnalogIn.SampleRate . . . . .	19-5
ATS2.AnalogIn.Source . . . . .	19-5

## Chapter 20

### Auxiliary Control Input and Output

ATS2.AuxControlIO.InputBitRdg . . . . .	20-1
ATS2.AuxControlIO.InputRdg . . . . .	20-2

ATS2.AuxControlIO.Output . . . . .	20-2
ATS2.AuxControlIO.OutputBit . . . . .	20-3

**Chapter 21**  
**Status Bits**

ATS2.Bits.AudioModeRdg . . . . .	21-1
ATS2.Bits.AuxBitsRdg . . . . .	21-2
ATS2.Bits.CategoryRdg . . . . .	21-3
ATS2.Bits.ChModeRdg . . . . .	21-5
ATS2.Bits.ChNumRdg . . . . .	21-7
ATS2.Bits.ClockAccuracyRdg . . . . .	21-9
ATS2.Bits.Cons.AudioMode . . . . .	21-11
ATS2.Bits.Cons.Category . . . . .	21-11
ATS2.Bits.Cons.Channels . . . . .	21-13
ATS2.Bits.Cons.ChNum . . . . .	21-13
ATS2.Bits.Cons.ClockAccuracy . . . . .	21-14
ATS2.Bits.Cons.Copyright . . . . .	21-15
ATS2.Bits.Cons.Emphasis . . . . .	21-15
ATS2.Bits.Cons.SampleFreq . . . . .	21-16
ATS2.Bits.Cons.SourceNum . . . . .	21-16
ATS2.Bits.CopyrightRdg . . . . .	21-18
ATS2.Bits.CrcRdg . . . . .	21-19
ATS2.Bits.DestinationRdg . . . . .	21-20
ATS2.Bits.EmphRdg . . . . .	21-21
ATS2.Bits.FlagRdg . . . . .	21-22
ATS2.Bits.FreqModeRdg . . . . .	21-24
ATS2.Bits.LocalAddressRdg . . . . .	21-25
ATS2.Bits.Mode . . . . .	21-25
ATS2.Bits.ModeRdg . . . . .	21-26
ATS2.Bits.OriginRdg . . . . .	21-26
ATS2.Bits.Pro.AudioMode . . . . .	21-27
ATS2.Bits.Pro.AuxBits . . . . .	21-28
ATS2.Bits.Pro.ChMode . . . . .	21-28
ATS2.Bits.Pro.CrcEnable . . . . .	21-29
ATS2.Bits.Pro.Destination . . . . .	21-30
ATS2.Bits.Pro.Emphasis . . . . .	21-30
ATS2.Bits.Pro.Flag . . . . .	21-30
ATS2.Bits.Pro.FreqMode . . . . .	21-31
ATS2.Bits.Pro.LocalAddress . . . . .	21-31
ATS2.Bits.Pro.LocalAddressAuto . . . . .	21-32
ATS2.Bits.Pro.Origin . . . . .	21-33
ATS2.Bits.Pro.RefSignal . . . . .	21-33
ATS2.Bits.Pro.SampleFreq . . . . .	21-34
ATS2.Bits.Pro.TimeOfDay . . . . .	21-35
ATS2.Bits.Pro.UserBits . . . . .	21-36
ATS2.Bits.Pro.WordLength . . . . .	21-36
ATS2.Bits.RefSignalRdg . . . . .	21-38

ATS2.Bits.SampleFreqRdg . . . . .	21-39
ATS2.Bits.SourceNumRdg . . . . .	21-42
ATS2.Bits.StatusXferToString . . . . .	21-45
ATS2.Bits.TimeOfDayRdg . . . . .	21-47
ATS2.Bits.UserBitsRdg . . . . .	21-48
ATS2.Bits.WordLengthRdg . . . . .	21-49
ATS2.Bits.XmitChannel . . . . .	21-51
ATS2.Bits.XmitStatus . . . . .	21-52

## Chapter 22

### DCX-127

ATS2.DCX.DcLevel . . . . .	22-1
ATS2.DCX.DcOutput . . . . .	22-1
ATS2.DCX.DigInFormat . . . . .	22-2
ATS2.DCX.DigInRdg . . . . .	22-3
ATS2.DCX.DigInRdgRate . . . . .	22-3
ATS2.DCX.DigInReady . . . . .	22-4
ATS2.DCX.DigInScale . . . . .	22-5
ATS2.DCX.DigInSettling . . . . .	22-5
ATS2.DCX.DigInTrig . . . . .	22-6
ATS2.DCX.DigOut . . . . .	22-6
ATS2.DCX.DigOutFormat . . . . .	22-6
ATS2.DCX.DigOutScale . . . . .	22-7
ATS2.DCX.DmmMode . . . . .	22-7
ATS2.DCX.DmmOffset . . . . .	22-9
ATS2.DCX.DmmRange . . . . .	22-9
ATS2.DCX.DmmRangeAuto . . . . .	22-10
ATS2.DCX.DmmRdg . . . . .	22-10
ATS2.DCX.DmmRdgRate . . . . .	22-11
ATS2.DCX.DmmReady . . . . .	22-11
ATS2.DCX.DmmScale . . . . .	22-12
ATS2.DCX.DmmSettling . . . . .	22-12
ATS2.DCX.DmmTrig . . . . .	22-13
ATS2.DCX.GateDelay . . . . .	22-13
ATS2.DCX.PortOutput . . . . .	22-14

## Chapter 23

### Digital Generator

ATS2.DGen.Ampl . . . . .	23-1
ATS2.DGen.BurstInterval . . . . .	23-2
ATS2.DGen.BurstLevel . . . . .	23-3
ATS2.DGen.BurstOnTime . . . . .	23-4
ATS2.DGen.ChBTrackA . . . . .	23-4
ATS2.DGen.DitherType . . . . .	23-5
ATS2.DGen.DualAmplRatio . . . . .	23-6
ATS2.DGen.EqAmpl . . . . .	23-6
ATS2.DGen.EqCurve . . . . .	23-7

ATS2.DGen.EqCurveColumn . . . . .	23-8
ATS2.DGen.EqCurveFilename . . . . .	23-8
ATS2.DGen.Freq . . . . .	23-9
ATS2.DGen.IMFreq . . . . .	23-9
ATS2.DGen.IMHighFreq . . . . .	23-10
ATS2.DGen.Invert . . . . .	23-10
ATS2.DGen.Offset . . . . .	23-11
ATS2.DGen.Output . . . . .	23-11
ATS2.DGen.OutputOn . . . . .	23-12
ATS2.DGen.Phase . . . . .	23-12
ATS2.DGen.RefdBr . . . . .	23-13
ATS2.DGen.RefFreq . . . . .	23-13
ATS2.DGen.RefVFS . . . . .	23-14
ATS2.DGen.StepRate . . . . .	23-14
ATS2.DGen.Wfm . . . . .	23-15
ATS2.DGen.WfmName . . . . .	23-16

## Chapter 24

### Digital Input/Output

ATS2.Dio.FlagInvalidRdg . . . . .	24-1
ATS2.Dio.FlagCodingRdg . . . . .	24-2
ATS2.Dio.FlagConfidenceRdg . . . . .	24-2
ATS2.Dio.FlagLockRdg . . . . .	24-3
ATS2.Dio.FlagParityRdg . . . . .	24-3
ATS2.Dio.InBitsDisplay . . . . .	24-4
ATS2.Dio.InDecode . . . . .	24-4
ATS2.Dio.InDeEmp . . . . .	24-5
ATS2.Dio.InFormat . . . . .	24-6
ATS2.Dio.InImpedance . . . . .	24-6
ATS2.Dio.InInput . . . . .	24-7
ATS2.Dio.InJitterBW . . . . .	24-7
ATS2.Dio.InJitterDetector . . . . .	24-8
ATS2.Dio.InJitterMode . . . . .	24-9
ATS2.Dio.InMonitorMode . . . . .	24-9
ATS2.Dio.InResolution . . . . .	24-10
ATS2.Dio.InScaleFreq . . . . .	24-10
ATS2.Dio.JitterRdg . . . . .	24-11
ATS2.Dio.JitterReady . . . . .	24-12
ATS2.Dio.JitterSettling . . . . .	24-12
ATS2.Dio.JitterTrig . . . . .	24-13
ATS2.Dio.OutEncode . . . . .	24-13
ATS2.Dio.OutFormat . . . . .	24-14
ATS2.Dio.OutInvert . . . . .	24-15
ATS2.Dio.OutJitterAmpl . . . . .	24-15
ATS2.Dio.OutJitterEqCurve . . . . .	24-16
ATS2.Dio.OutJitterFreq . . . . .	24-16
ATS2.Dio.OutJitterType . . . . .	24-17



ATS2.Dio.OutParityError . . . . .	24-17
ATS2.Dio.OutPreEmp . . . . .	24-17
ATS2.Dio.OutRate . . . . .	24-19
ATS2.Dio.OutResolution . . . . .	24-19
ATS2.Dio.OutScaleFreq . . . . .	24-19
ATS2.Dio.OutSendInvalid . . . . .	24-20
ATS2.Dio.OutVoltage . . . . .	24-21
ATS2.Dio.PeakRdg . . . . .	24-21
ATS2.Dio.PeakReady . . . . .	24-22
ATS2.Dio.PeakTrig . . . . .	24-23
ATS2.Dio.RateRdg . . . . .	24-23
ATS2.Dio.RateReady . . . . .	24-24
ATS2.Dio.RateSettling . . . . .	24-25
ATS2.Dio.RateTrig . . . . .	24-25
ATS2.Dio.RefRate . . . . .	24-26
ATS2.Dio.VoltageRdg . . . . .	24-26
ATS2.Dio.VoltageReady . . . . .	24-27
ATS2.Dio.VoltageSettling . . . . .	24-27
ATS2.Dio.VoltageTrig . . . . .	24-28

## Chapter 25

### Hardware

ATS2.Hardware.AdjustmentDate . . . . .	25-1
ATS2.Hardware.BoardSerialNumber . . . . .	25-2
ATS2.Hardware.ManufactureDate . . . . .	25-2
ATS2.Hardware.OptionInstalled . . . . .	25-3
ATS2.Hardware.SerialNumber . . . . .	25-3

## Chapter 26

### Audio Analyzer

ATS2.Inst.Analyzer.DCCoupled . . . . .	26-1
ATS2.Inst.Analyzer.FreqRdg . . . . .	26-1
ATS2.Inst.Analyzer.FreqReady . . . . .	26-2
ATS2.Inst.Analyzer.FreqSettling . . . . .	26-3
ATS2.Inst.Analyzer.FreqTrig . . . . .	26-4
ATS2.Inst.Analyzer.FuncBPBRFreq . . . . .	26-4
ATS2.Inst.Analyzer.FuncBPBRTuning . . . . .	26-6
ATS2.Inst.Analyzer.FuncDetector . . . . .	26-7
ATS2.Inst.Analyzer.FuncFilter . . . . .	26-8
ATS2.Inst.Analyzer.FuncFilterFilename . . . . .	26-10
ATS2.Inst.Analyzer.FuncFilterHP . . . . .	26-10
ATS2.Inst.Analyzer.FuncFilterLP . . . . .	26-11
ATS2.Inst.Analyzer.FuncFilterUserDefined . . . . .	26-11
ATS2.Inst.Analyzer.FuncMode . . . . .	26-12
ATS2.Inst.Analyzer.FuncPhaseMode . . . . .	26-15
ATS2.Inst.Analyzer.FuncRange . . . . .	26-16
ATS2.Inst.Analyzer.FuncRangeAuto . . . . .	26-17

ATS2.Inst.Analyzer.FuncRdg . . . . .	26-17
ATS2.Inst.Analyzer.FuncReady . . . . .	26-18
ATS2.Inst.Analyzer.FuncSettling . . . . .	26-19
ATS2.Inst.Analyzer.FuncTrig . . . . .	26-19
ATS2.Inst.Analyzer.InputFormat . . . . .	26-20
ATS2.Inst.Analyzer.LevelRdg . . . . .	26-20
ATS2.Inst.Analyzer.LevelReady . . . . .	26-21
ATS2.Inst.Analyzer.LevelSettling . . . . .	26-22
ATS2.Inst.Analyzer.LevelTrig . . . . .	26-22
ATS2.Inst.Analyzer.Range . . . . .	26-23
ATS2.Inst.Analyzer.RangeAuto . . . . .	26-23
ATS2.Inst.Analyzer.RdgRate . . . . .	26-24

## Chapter 27

### Multitone Audio Analyzer

ATS2.Inst.FastTest.FreqRes . . . . .	27-1
ATS2.Inst.FastTest.InputFormat . . . . .	27-2
ATS2.Inst.FastTest.Mode . . . . .	27-2
ATS2.Inst.FastTest.PhaseDisplay . . . . .	27-6
ATS2.Inst.FastTest.Processing . . . . .	27-7
ATS2.Inst.FastTest.Rdg . . . . .	27-8
ATS2.Inst.FastTest.Ready . . . . .	27-9
ATS2.Inst.FastTest.TransformLength . . . . .	27-10
ATS2.Inst.FastTest.Trig . . . . .	27-12
ATS2.Inst.FastTest.TrigDelay . . . . .	27-12
ATS2.Inst.FastTest.TrigSource . . . . .	27-13

## Chapter 28

### FFT Spectrum Analyzer

ATS2.Inst.FFT.AcquireLength . . . . .	28-1
ATS2.Inst.FFT.Averages . . . . .	28-2
ATS2.Inst.FFT.AveragesType . . . . .	28-4
ATS2.Inst.FFT.Rdg . . . . .	28-5
ATS2.Inst.FFT.Ready . . . . .	28-6
ATS2.Inst.FFT.Trig . . . . .	28-7
ATS2.Inst.FFT.InputFormat . . . . .	28-7
ATS2.Inst.FFT.StartTime . . . . .	28-8
ATS2.Inst.FFT.SubtractDC . . . . .	28-9
ATS2.Inst.FFT.TransformLength . . . . .	28-9
ATS2.Inst.FFT.TrigDelay . . . . .	28-10
ATS2.Inst.FFT.TrigLevel . . . . .	28-11
ATS2.Inst.FFT.TrigPolarity . . . . .	28-11
ATS2.Inst.FFT.TrigSensitivity . . . . .	28-12
ATS2.Inst.FFT.TrigSource . . . . .	28-12
ATS2.Inst.FFT.WfmDisplay . . . . .	28-14
ATS2.Inst.FFT.Window . . . . .	28-16

---

## Chapter 29

### Harmonic Distortion Analyzer

ATS2.Inst.Harmonic.AmplRdg . . . . .	29-1
ATS2.Inst.Harmonic.AmplReady . . . . .	29-2
ATS2.Inst.Harmonic.AmplSettling . . . . .	29-3
ATS2.Inst.Harmonic.AmplTrig . . . . .	29-3
ATS2.Inst.Harmonic.Freq . . . . .	29-4
ATS2.Inst.Harmonic.FreqRdg . . . . .	29-5
ATS2.Inst.Harmonic.FreqReady . . . . .	29-5
ATS2.Inst.Harmonic.FreqSettling . . . . .	29-6
ATS2.Inst.Harmonic.FreqTrig . . . . .	29-7
ATS2.Inst.Harmonic.Harmonics . . . . .	29-7
ATS2.Inst.Harmonic.InputFormat . . . . .	29-8
ATS2.Inst.Harmonic.Rdg . . . . .	29-9
ATS2.Inst.Harmonic.Ready . . . . .	29-10
ATS2.Inst.Harmonic.RatioUnits . . . . .	29-11
ATS2.Inst.Harmonic.Selectivity . . . . .	29-12
ATS2.Inst.Harmonic.Settling . . . . .	29-12
ATS2.Inst.Harmonic.Trig . . . . .	29-13
ATS2.Inst.Harmonic.Tuning . . . . .	29-13

## Chapter 30

### INTERVU Digital Interface Analyzer

ATS2.Inst.Intervu.AcquisitionPosition . . . . .	30-1
ATS2.Inst.Intervu.Averages . . . . .	30-1
ATS2.Inst.Intervu.JitterDetection . . . . .	30-2
ATS2.Inst.Intervu.TrigPolarity . . . . .	30-5
ATS2.Inst.Intervu.WfmDisplay . . . . .	30-6
ATS2.Inst.Intervu.Window . . . . .	30-7

## Chapter 31

### Analyzer Instrument Selection and Reference

ATS2.Inst.RefdBm . . . . .	31-1
ATS2.Inst.RefdBr . . . . .	31-2
ATS2.Inst.RefdBrAuto . . . . .	31-3
ATS2.Inst.RefFreq . . . . .	31-4
ATS2.Inst.RefFreqAuto . . . . .	31-5
ATS2.Inst.RefVFS . . . . .	31-6
ATS2.Inst.RefWatts . . . . .	31-6
ATS2.Inst.Selection . . . . .	31-7

## Chapter 32

### Speaker

ATS2.Speaker.Mute . . . . .	32-1
ATS2.Speaker.Source . . . . .	32-2
ATS2.Speaker.Volume . . . . .	32-2

---

## Chapter 33 Switcher

ATS2.SWR.In . . . . .	33-1
ATS2.SWR.InOut . . . . .	33-1
ATS2.SWR.Mode . . . . .	33-2
ATS2.SWR.Offset . . . . .	33-4
ATS2.SWR.Out . . . . .	33-4

## Chapter 34 Sync/Ref Input

ATS2.Sync.FrameLock . . . . .	34-1
ATS2.Sync.Freq . . . . .	34-1
ATS2.Sync.FreqRdg . . . . .	34-2
ATS2.Sync.FreqReady . . . . .	34-3
ATS2.Sync.FreqTrig . . . . .	34-4
ATS2.Sync.Impedance . . . . .	34-4
ATS2.Sync.OutOfRangeRdg . . . . .	34-5
ATS2.Sync.Source . . . . .	34-5
ATS2.Sync.SourceInput . . . . .	34-5

## Chapter 35 Main Triggers

ATS2.Trigger.Coding . . . . .	35-1
ATS2.Trigger.Confidence . . . . .	35-1
ATS2.Trigger.Lock . . . . .	35-2
ATS2.Trigger.Parity . . . . .	35-2
ATS2.Trigger.Source . . . . .	35-4
ATS2.Trigger.Source . . . . .	35-6

## Appendix A Settling Algorithm

Description . . . . .	A-1
Settling Parameter Descriptions . . . . .	A-1

## Appendix B Parameter ID# List . . . . .

B-1

## Appendix C FFT Window Descriptions. . . . .

C-1

# Chapter 1

## ATS Extensions Reference

### Introduction

This chapter of the manual is divided into three sections.

The first section consists of ATS system panels listed alphabetically by panel title. Each page consists of one or more panels and the commands applicable to each panel.

The second section consists of an alphabetical listing of all AP Basic extensions.

The third section consists of technical reference information for the command extensions sorted alphabetically. Each command contains many of the following parts.

<b>Name</b>	<b>Description</b>
<i>Syntax:</i>	Programming usage information.
<i>Command type:</i>	Method or Property
<i>Data type:</i>	Setting Data Type.
<i>Result:</i>	Query Data Type.
<i>Description:</i>	Technical Information.
<i>See also:</i>	Commands related to the current command that may contain relevant information.
<i>Example:</i>	Example procedure/macro
<i>Example Output:</i>	When an example program produces output to the immediate window of the Procedure/Macro Editor or output to a file a sample of what the output will be shown in this location.
<i>Comments:</i>	Additional information relating to the example procedure/macro.

### Manual Conventions

---

This manual uses the following typographic conventions.

Example	Description
<i>event,</i> <i>var, arg</i>	For the syntax part of each command, italicized words indicate place holders where the user must enter additional information.
FILENAME.TXT	Words in all CAPITOL letters indicate file names.
Sub Main ATS2.AGen.Amp = 1.0 End Sub	This font is used in all example macros and code modules.
<i>[expression list]</i>	In syntax, items inside square brackets are optional.
<i>{While   Until}</i>	In syntax, braces and a vertical bar indicate a choice between two or more items.
<b>Command</b>	For the syntax part of each command, the bold characters identify the part of the command that must be entered.
ATS.Prompt. _ Text "Example"	The line continue character ( _ ) is used to indicate that the code from one line to the next should be typed on one line.

# Chapter 2

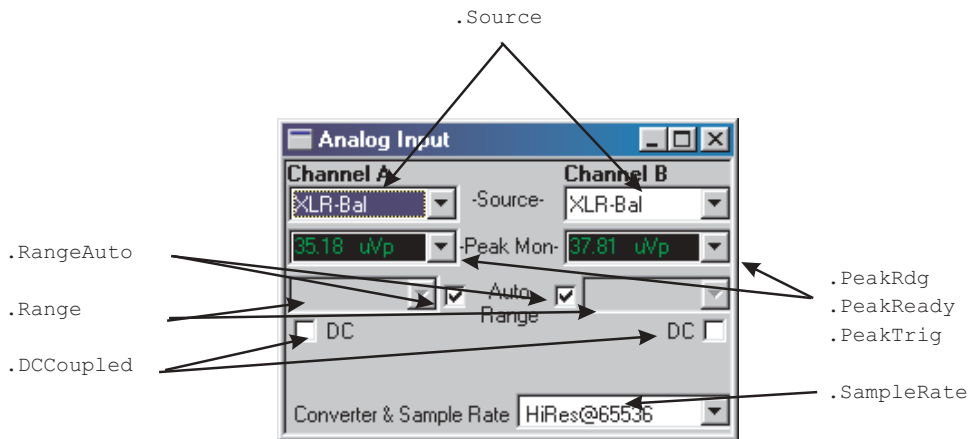
## System Panels

### Analog Input

All commands on this page start with the following:

**ATS2.AnalogIn**

Example: `ATS2.AnalogIn.DCCoupled`

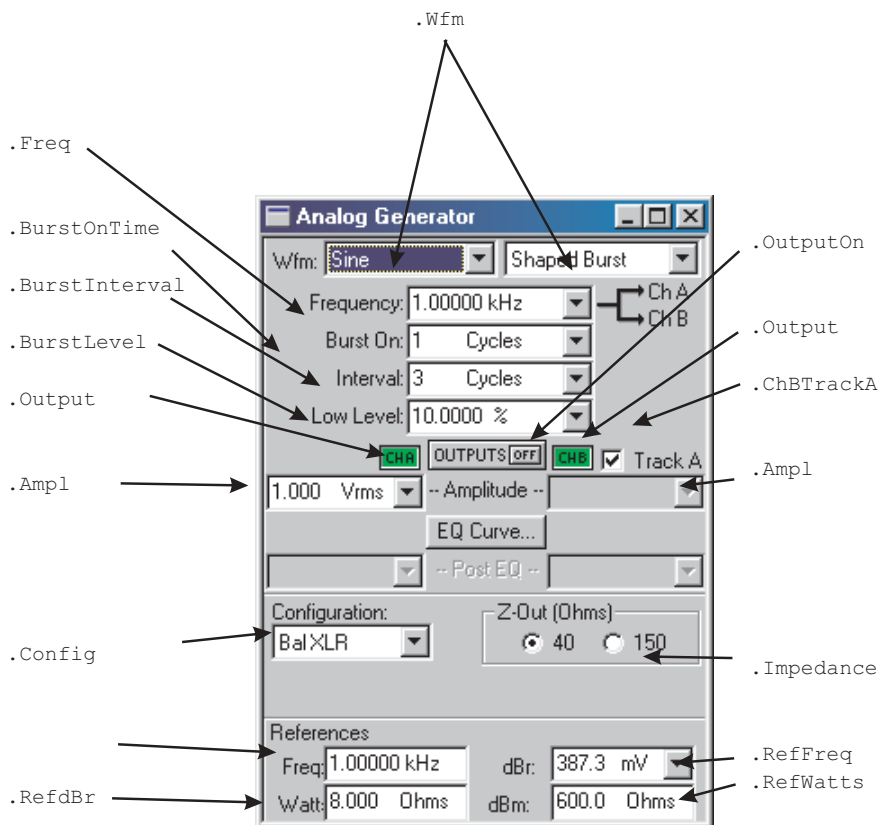


### Analog Generator ...

All commands on this page start with the following:

**ATS2.AGen**

Example: `ATS2.AGen.Freq`



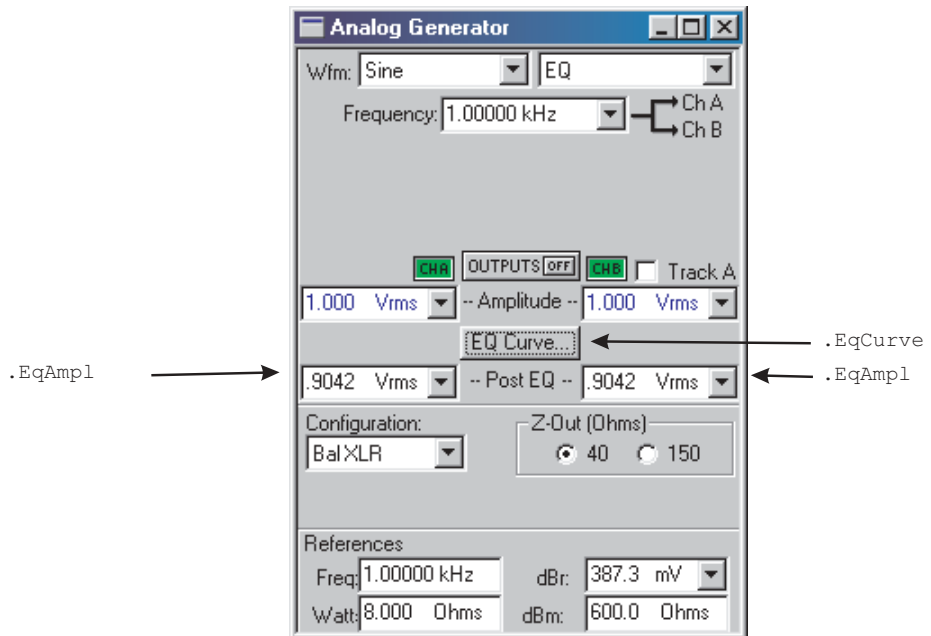


## Analog Generator Continued ...

All commands on this page start with the following:

**ATS2 .AGen**

Example: `ATS2.AGen.EqAmp1`

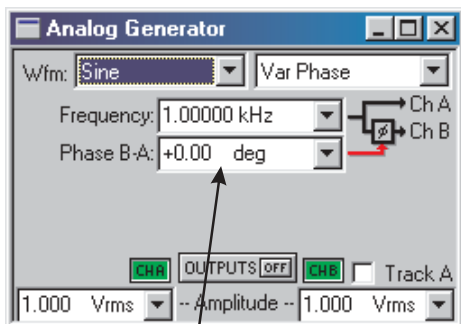


## Analog Generator Continued

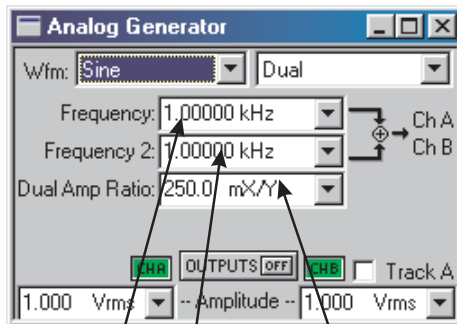
All commands on this page start with the following:

**ATS2.AGen**

Example: ATS2.AGen.Phase



.Phase



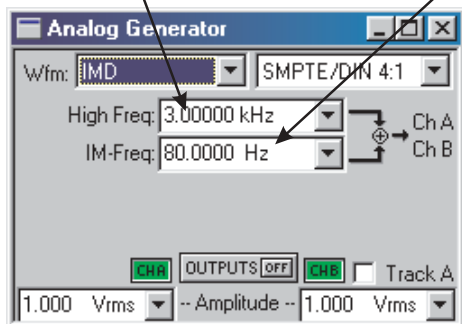
.Freq

.Freq

.DualAmplRatio

.IMHighFreq

.IMFreq

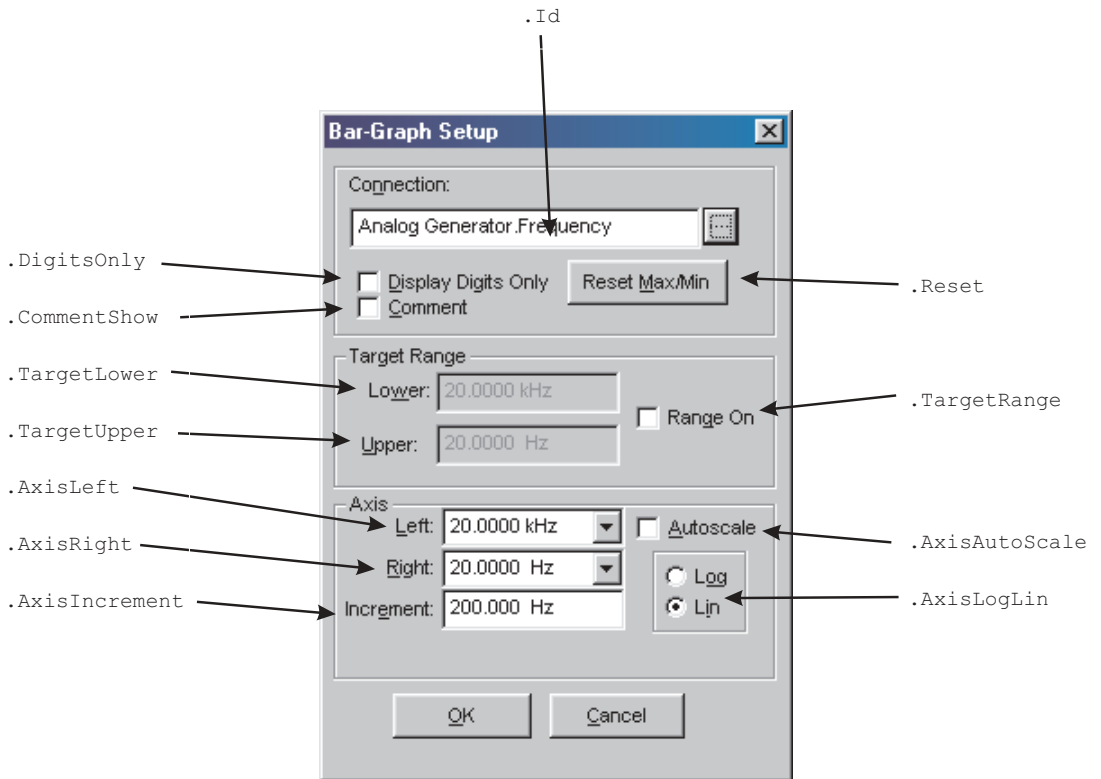


## BarGraph

All commands on this page start with the following:

**ATS.Bar**

Example: `ATS.Bar.DigitsOnly`

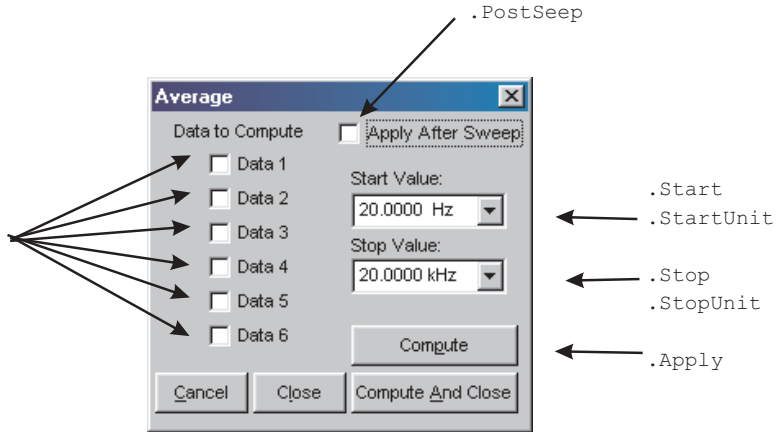


### Computes ...

All commands for the top diagram start with the following:

**ATS.Compute.Avg**

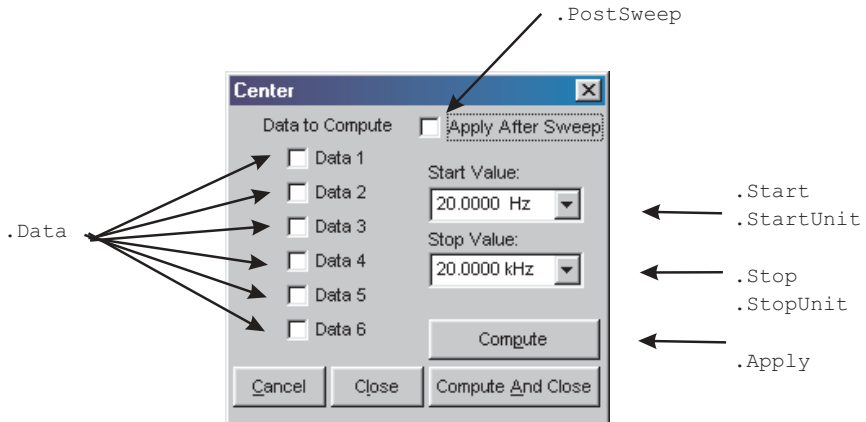
Example: `ATS.Compute.Avg.Apply`



All commands for the bottom diagram start with the following:

**ATS.Compute.Center**

Example: `ATS.Compute.Center.Apply`

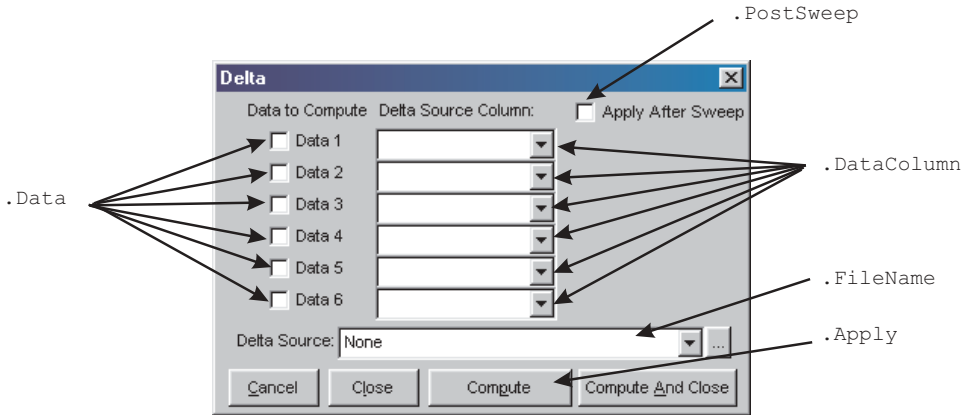


## Computes Continued ...

All commands for the top diagram start with the following:

**ATS.Compute.Delta**

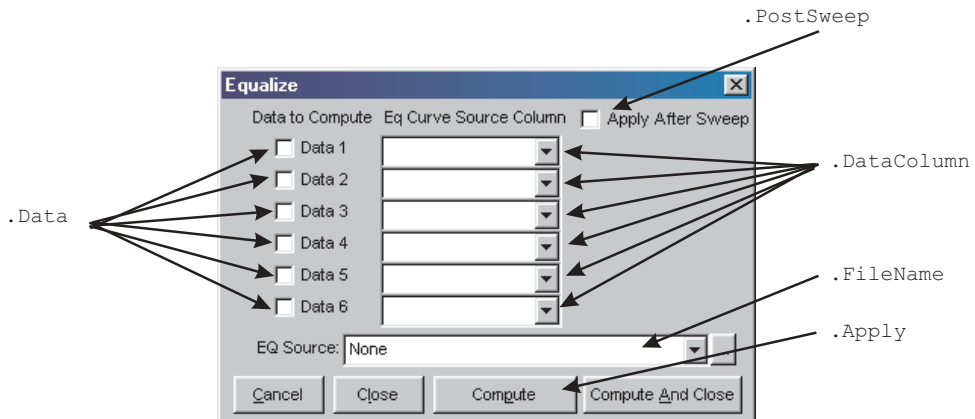
Example: `ATS.Compute.Delta.Apply`



All commands for the bottom diagram start with the following:

**ATS.Compute.Equalize**

Example: `ATS.Compute.Equalize.Apply`

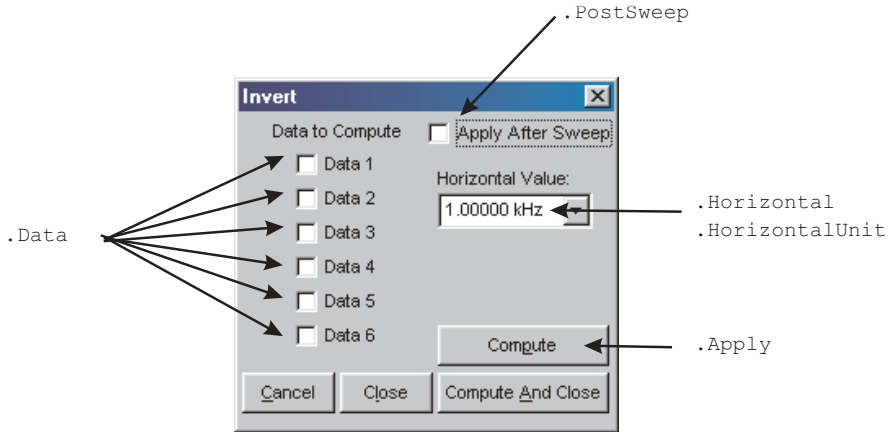


### Computes Continued ...

All commands for the top diagram start with the following:

**ATS.Compute.Invert**

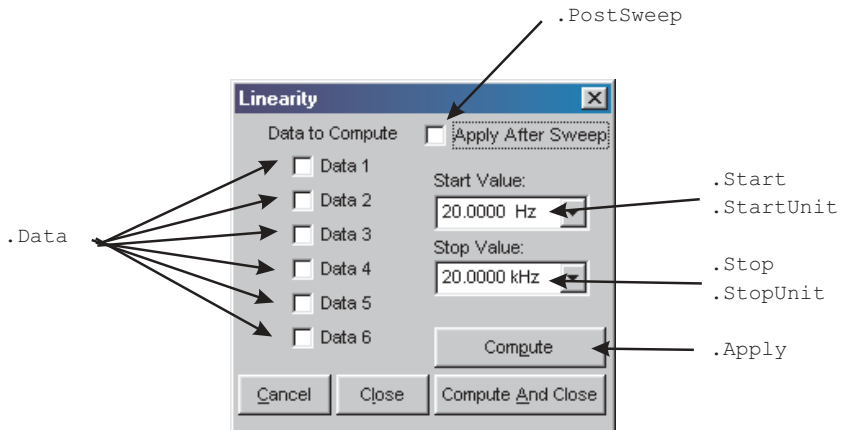
Example: `ATS.Compute.Invert.Apply`



All commands for the bottom diagram start with the following:

**ATS.Compute.Linearity**

Example: `ATS.Compute.Linearity.Apply`

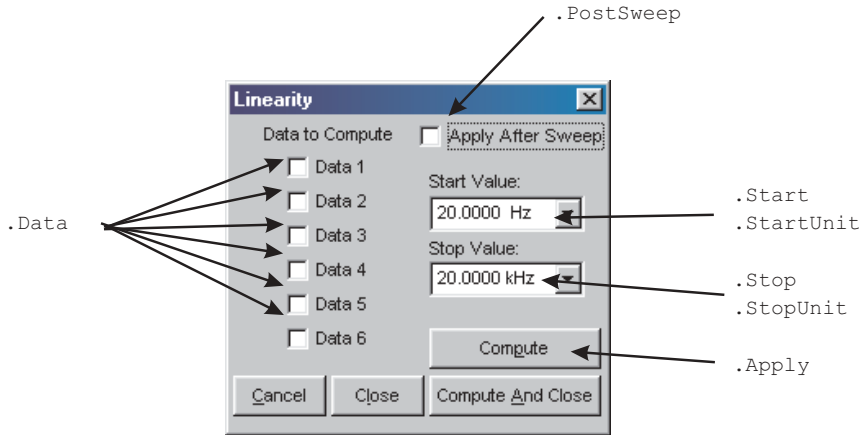


## Computes Continued ...

All commands for the top diagram start with the following:

### ATS.Compute.Max

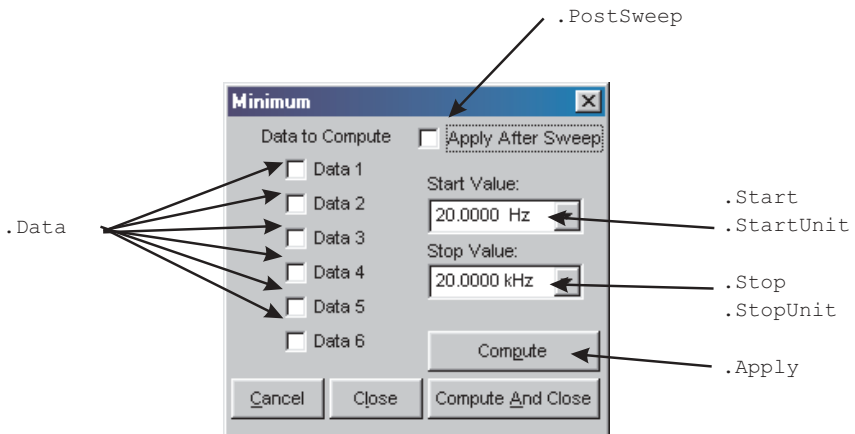
Example: ATS.Compute.Max.Apply



All commands for the bottom diagram start with the following:

### ATS.Compute.Min

Example: ATS.Compute.Min.Apply

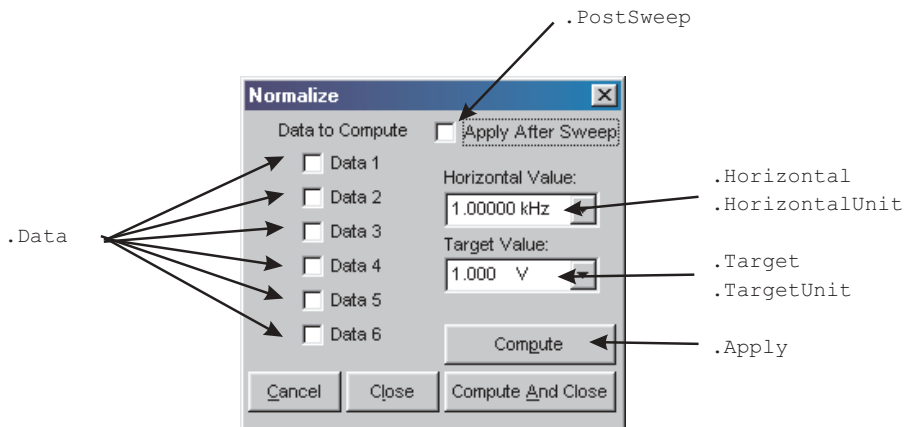


### Computes Continued ...

All commands for the top diagram start with the following:

**ATS.Compute.Normalize**

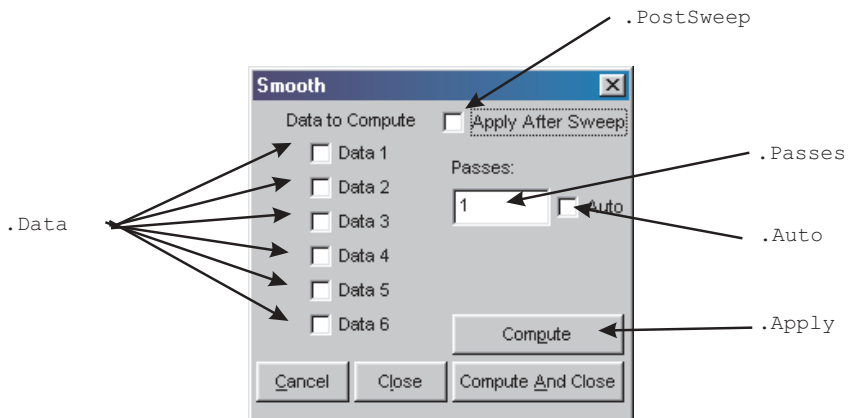
Example: `ATS.Compute.Normalize.Apply`



All commands for the bottom diagram start with the following:

**ATS.Compute.Smooth**

Example: `ATS.Compute.Smooth.Apply`





## DCX-127

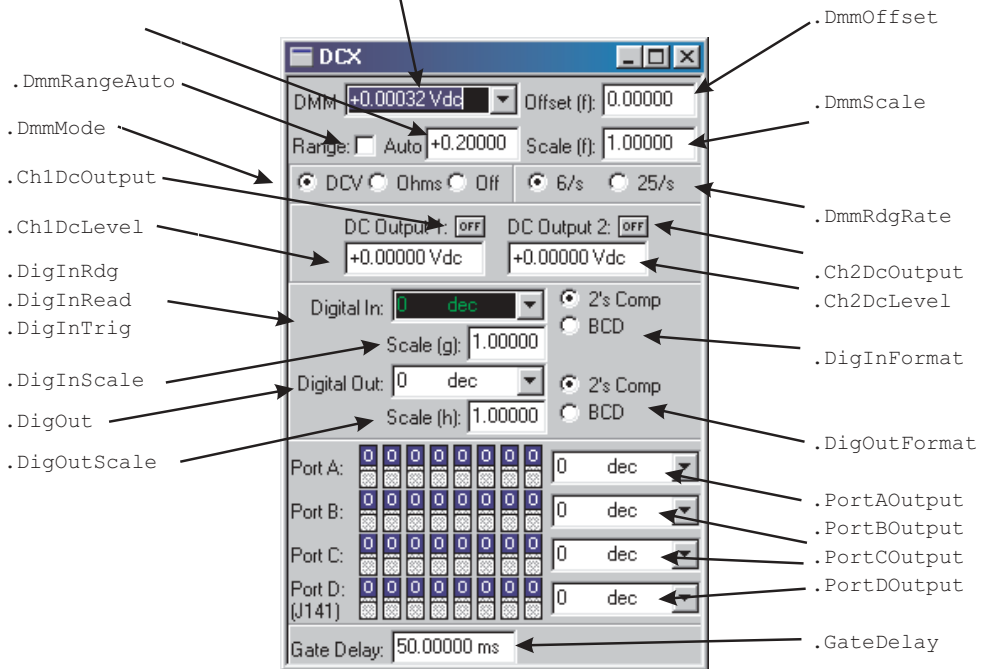
All commands on this page start with the following:

**ATS2.DCX**

Example: ATS2.DCX.DigOut

.DmmRange

.DmmRdg  
.DmmReady  
.DmmTrig



## Digital Analyzer Panels

The screenshot shows the 'Analyzer' window with the following settings:

- Instrument: Audio Analyzer
- Ch A Input: Analog
- Ch B
- Level: 8.423  $\mu$ V (Ch A), 9.077  $\mu$ V (Ch B)
- Freq: 9.63150 kHz (Ch A), 9.44758 kHz (Ch B)
- Function: 8.411  $\mu$ V (Ch A), 9.073  $\mu$ V (Ch B)
- Measurement Function: Amplitude
- Det: Auto, RMS, BP/BR Fltr Freq
- BW: < 10 Hz, Fs/2
- Fltr: None
- References:
  - dBr A: 387.3 mV
  - dBr B: 387.3 mV
  - Freq: 1.00000 kHz
  - V/FS: 1.000 V
  - Watts: 8.000 Ohms
  - dBm: 600.0 Ohms

Annotations with arrows point to the following elements:

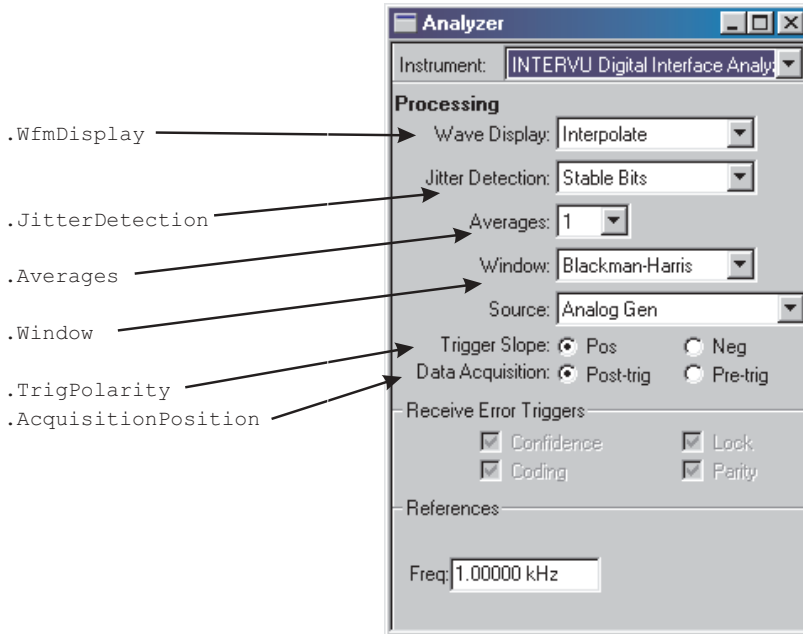
- ATS2.Inst.Selection points to the Instrument dropdown.
- ATS2.Inst.RefdBr points to the dBr A and dBr B reference values.
- ATS2.Inst.RefFreq points to the Freq reference value.
- ATS2.Inst.RefVFS points to the V/FS reference value.
- ATS2.Inst.RefdBm points to the dBm reference value.

## Digital Interface Analyzer (INTERVU)

All commands on this page start with the following:

**ATS2.Inst.Intervu**

Example: `ATS2.Inst.Intervu.WfmDisplay`

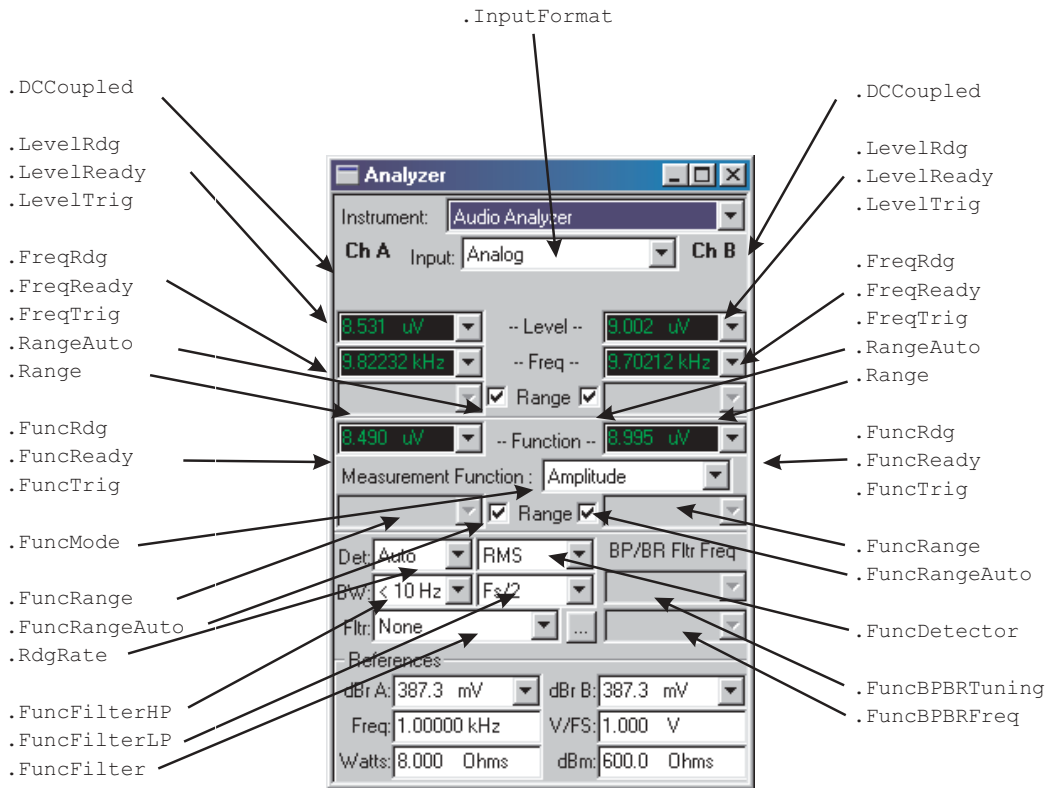


## Audio Analyzer (ANALYZER)

All commands on this page start with the following:

**ATS2.Inst.Analyzer**

Example: `ATS2.Inst.Analyzer.LevelRdg`

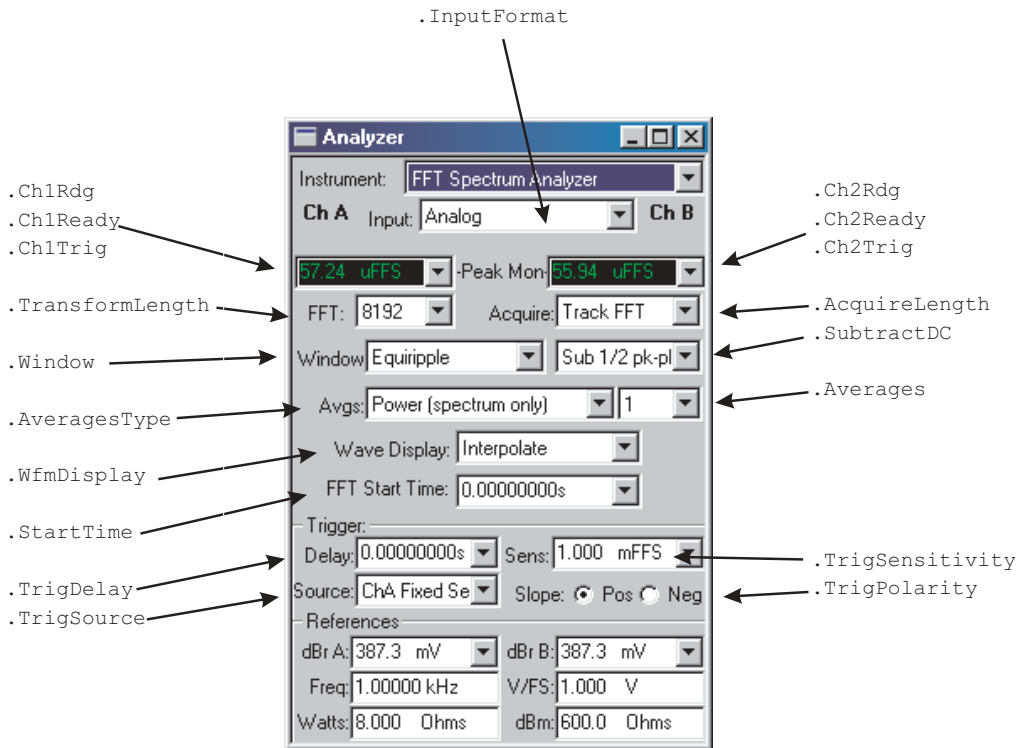


## FFT Spectrum Analyzer (FFT)

All commands on this page start with the following:

**AST2.Inst.FFT**

Example: `ATS2.Inst.FFT.InputFormat`

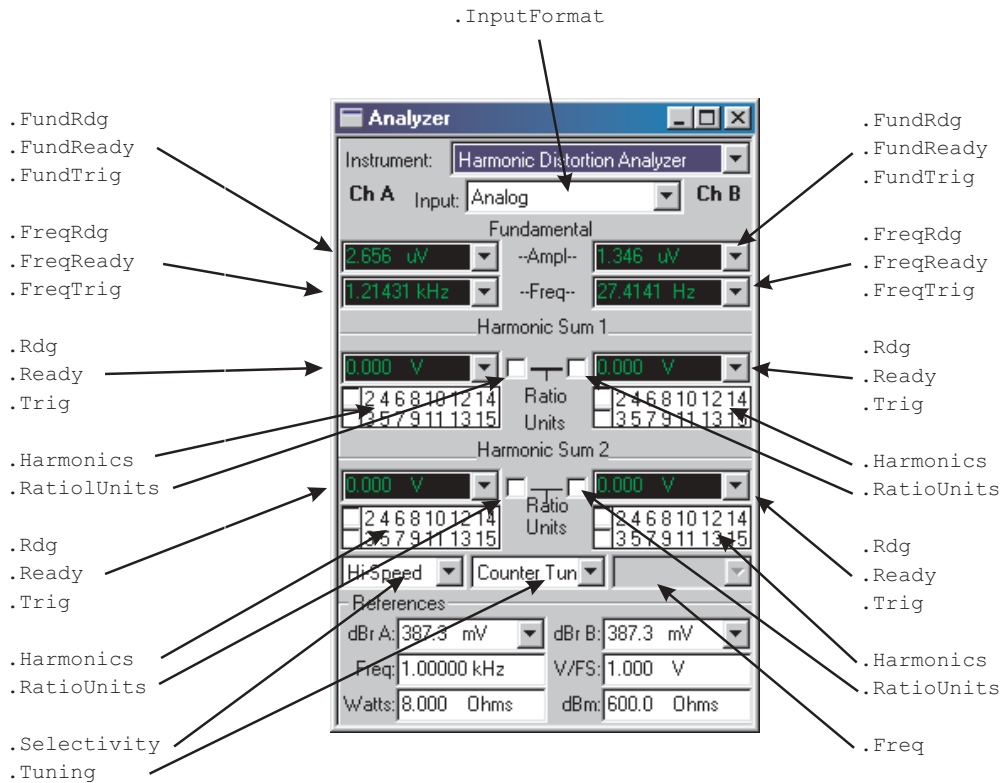


## Harmonic Distortion Analyzer (DISTORT)

All commands on this page start with the following:

**ATS2.Inst.Harmonic**

Example: `ATS2.Inst.Harmonic.InputFormat`

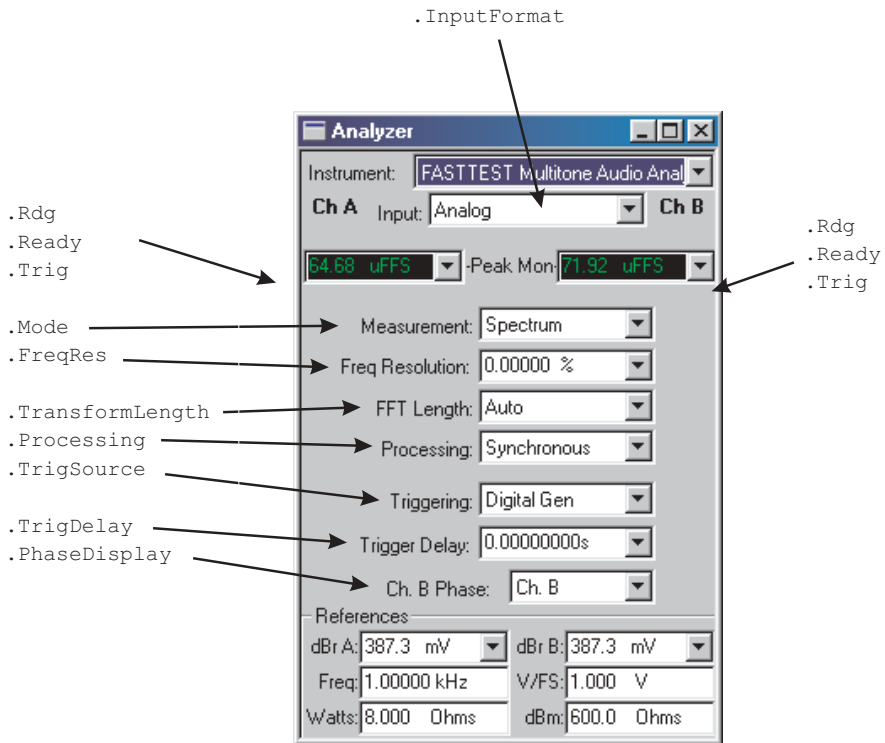


## Multitone Audio Analyzer (FASTTEST)

All commands on this page start with the following:

**ATS2.Inst.FastTest**

Example: `ATS2.Inst.FastTest.InputFormat`

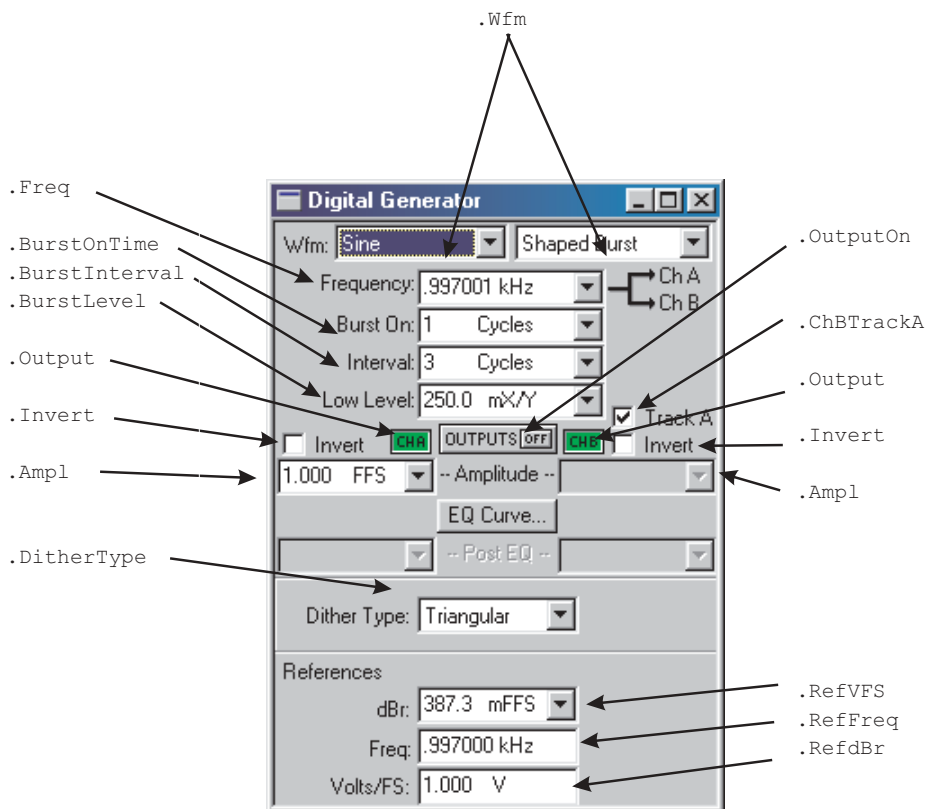


### Digital Generator ...

All commands on this page start with the following:

**ATS2.DGen**

Example: ATS2.DGen.Freq



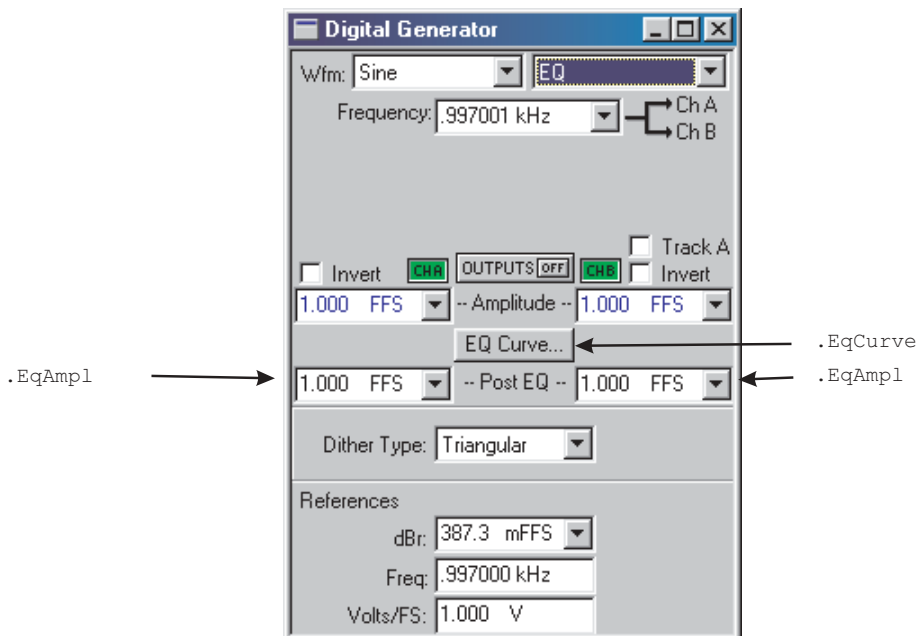


## Digital Generator Continued ...

All commands on this page start with the following:

**ATS2.DGen**

Example: `ATS2.DGen.EqAmpl`

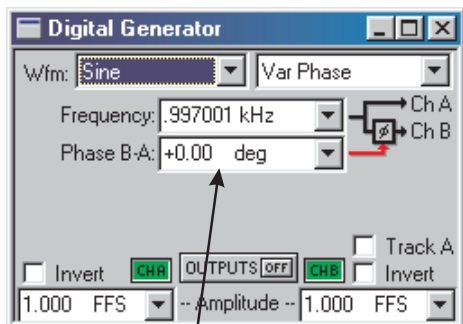


### Digital Generator Continued ...

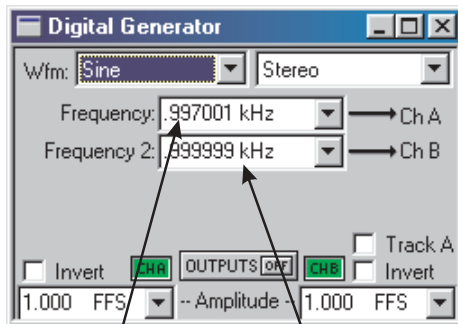
All commands on this page start with the following:

**ATS2.DGen**

Example: ATS2.DGen.Phase

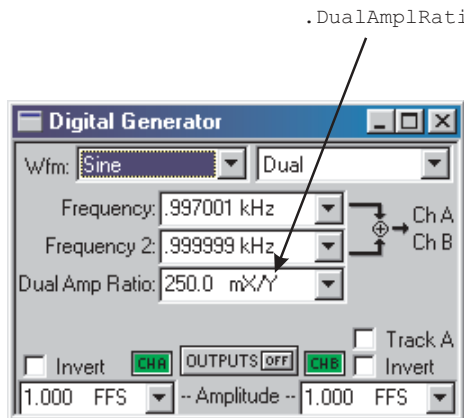


.Phase

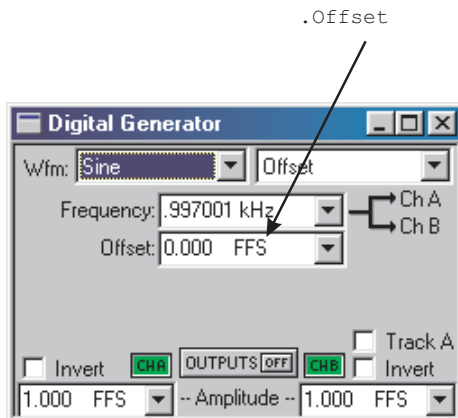


.Freq

.Freq



.DualAmplRatio



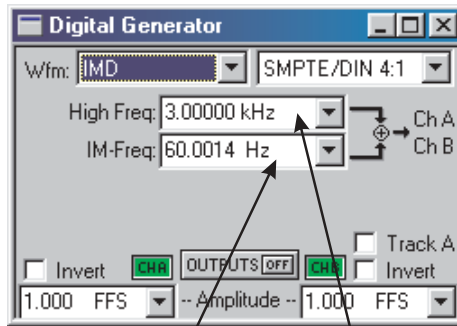
.Offset

## Digital Generator Continued

All commands on this page start with the following:

**ATS2.DGen**

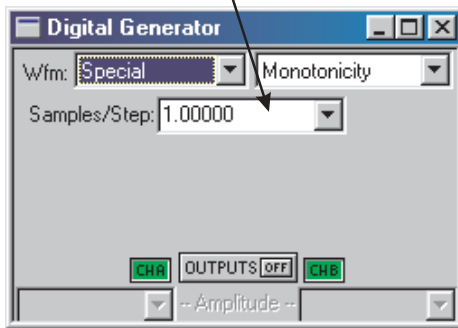
Example: ATS2.DGen.IMFreq



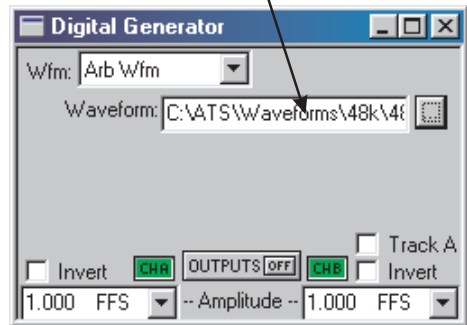
.IMFreq

.IMHighFreq

.StepRate



.WfmName

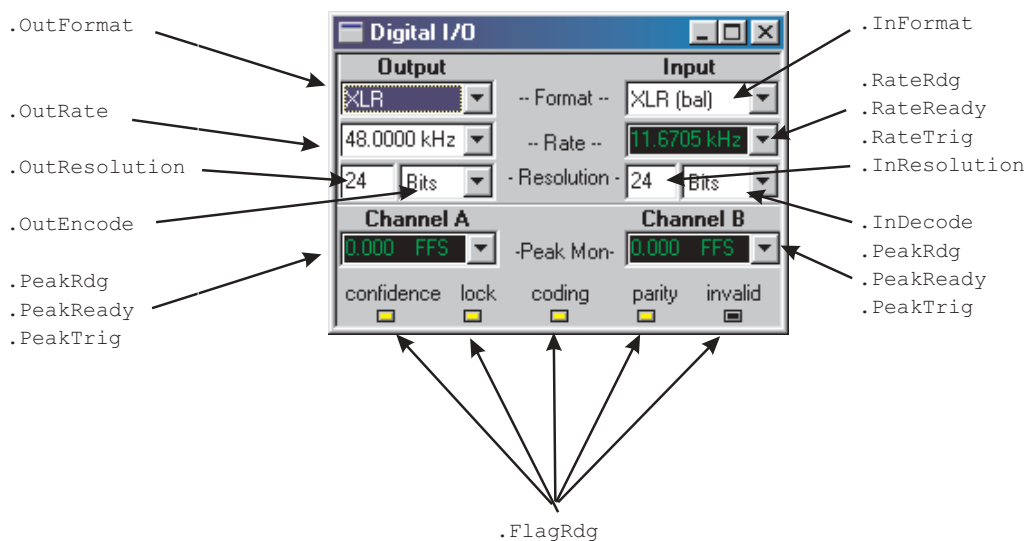


### Digital IO Parameters - Input/Output small panel view...

All commands on this page start with the following:

**ATS2.Dio**

Example: ATS2.Dio.OutFormat

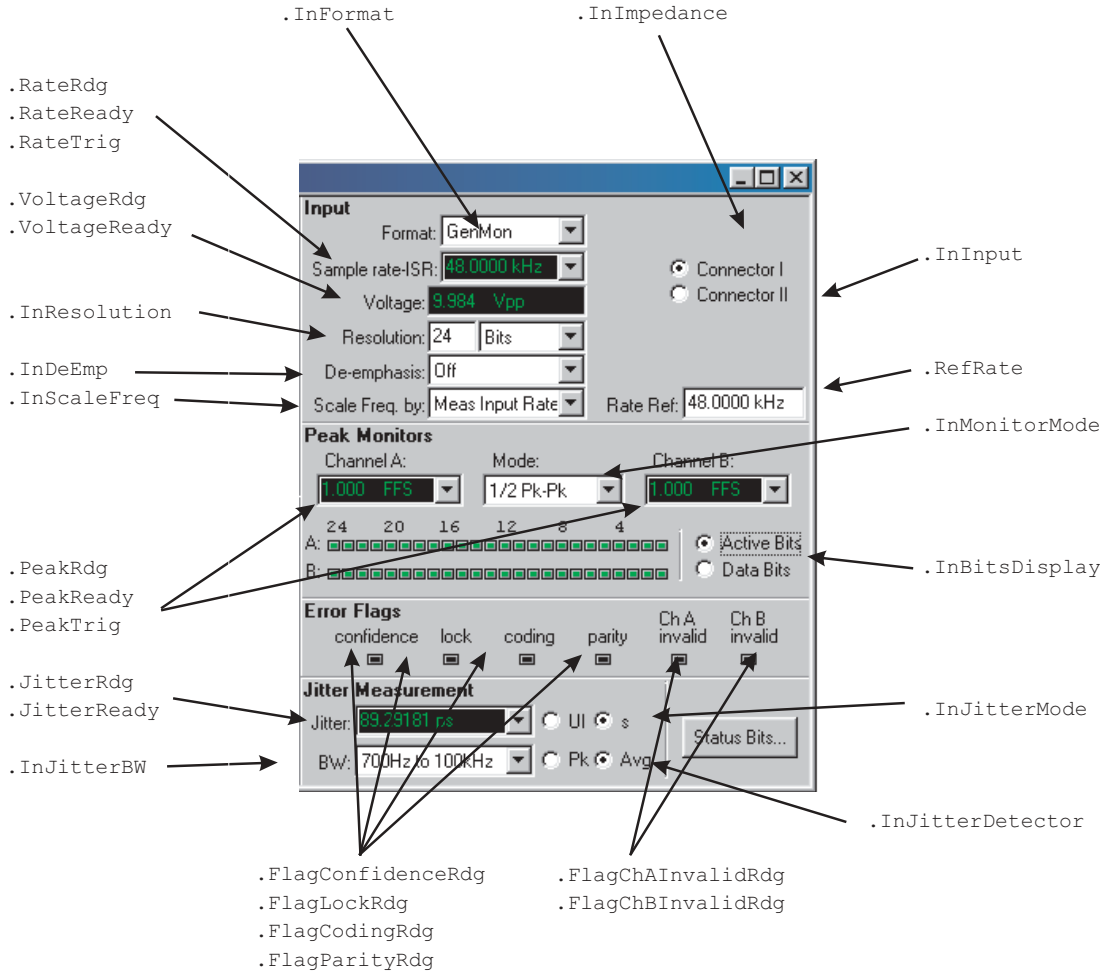


## Digital IO Parameters - Input ...

All commands on this page start with the following:

**ATS2.Dio**

Example: `ATS2.Dio.InFormat`

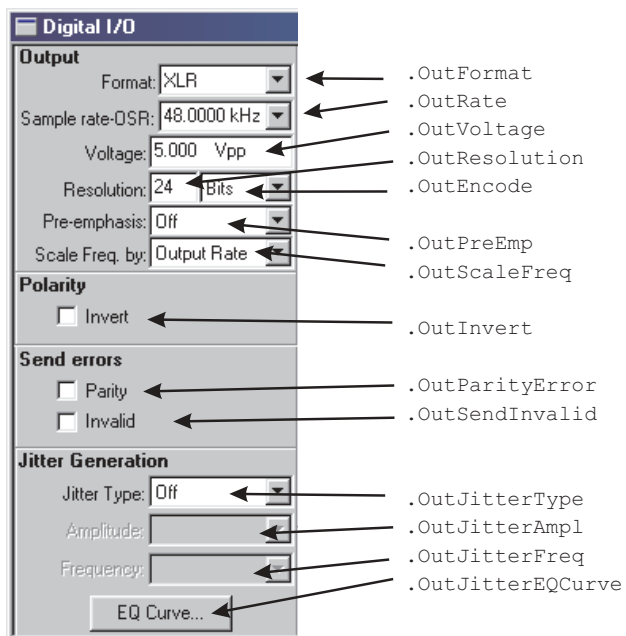


### Digital IO Parameters - Output Continued

All commands on this page start with the following:

**ATS2.Dio**

Example: `ATS2.Dio.OutFormat`

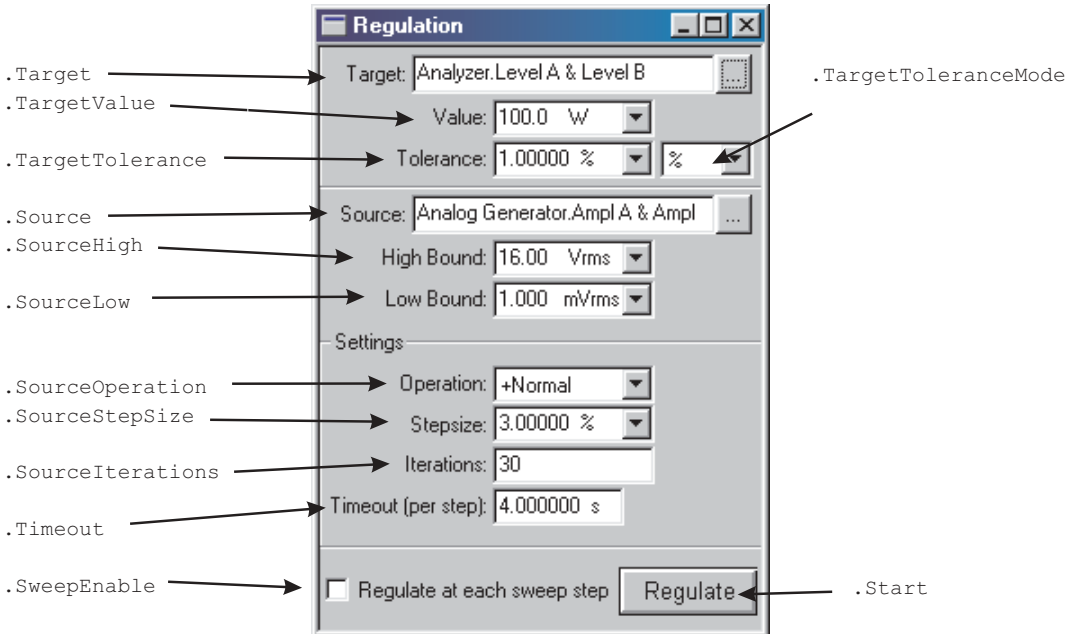


## Regulation

All commands on this page start with the following:

**AST2.Reg**

Example: `ATS2.Reg.Target`



## Settling

	Tolerance	Floor	Points	Delay	Algorithm	
ATS2.Inst.Analyzer.LevelSettling	Analyzer.Level A	1.00000 %	1.000 uV	3	30.00000 ms	Flat
ATS2.Inst.Analyzer.LevelSettling	Analyzer.Level B	1.00000 %	1.000 uV	3	30.00000 ms	Flat
ATS2.Inst.Analyzer.FreqSettling	Analyzer.Frequency A	0.50000 %	10.0000 mHz	1	2.000000 ms	Flat
ATS2.Inst.Analyzer.FreqSettling	Analyzer.Frequency B	0.50000 %	10.0000 mHz	1	2.000000 ms	Flat
ATS2.Inst.Analyzer.FuncSettling	Analyzer.Amplitude A	1.00000 %	1.000 uV	3	30.00000 ms	Flat
ATS2.Inst.Analyzer.FuncSettling	Analyzer.Amplitude B	1.00000 %	1.000 uV	3	30.00000 ms	Flat

	Tolerance	Floor	Points	Delay	Algorithm	
ATS2.Inst.Harmonic.AmplSettling	Harmonic.ChA Fund Ampl	1.00000 %	1.000 uV	3	30.00000 ms	Flat
ATS2.Inst.Harmonic.AmplSettling	Harmonic.ChB Fund Ampl	1.00000 %	1.000 uV	3	30.00000 ms	Flat
ATS2.Inst.Harmonic.FreqSettling	Harmonic.ChA Fund Freq	0.50000 %	10.0000 mHz	1	2.000000 ms	Flat
ATS2.Inst.Harmonic.FreqSettling	Harmonic.ChB Fund Freq	0.50000 %	10.0000 mHz	1	2.000000 ms	Flat
ATS2.Inst.Harmonic.Settling	Harmonic.ChA Harm Sum1	0.50000 %	10.00 mV	1	2.000000 ms	Flat
ATS2.Inst.Harmonic.Settling	Harmonic.ChA Harm Sum2	0.50000 %	10.00 mV	1	2.000000 ms	Flat
ATS2.Inst.Harmonic.Settling	Harmonic.ChB Harm Sum1	0.50000 %	10.00 mV	1	2.000000 ms	Flat
ATS2.Inst.Harmonic.Settling	Harmonic.ChB Harm Sum2	0.50000 %	10.00 mV	1	2.000000 ms	Flat

	Tolerance	Floor	Points	Delay	Algorithm	
ATS2.Dio.RateSettling	Digital I/O Sample Rate:	0.50000 %	100.000 mHz	3	30.00000 ms	Flat
ATS2.Dio.VoltageSettling	Voltage:	3.00000 %	10.00 mVpp	3	30.00000 ms	Flat
ATS2.Dio.JitterSettling	Jitter Amplitude:	3.00000 %	3.000000 ns	3	100.0000 ms	Exponential
ATS2.Dcx.DmmSettling	DCV	0.20000 %	+0.00050 Vdc	3	30.00000 ms	Flat
ATS2.Dcx.DigInSettling	Digital In:	0.00000 %	0 dec	3	30.00000 ms	Flat

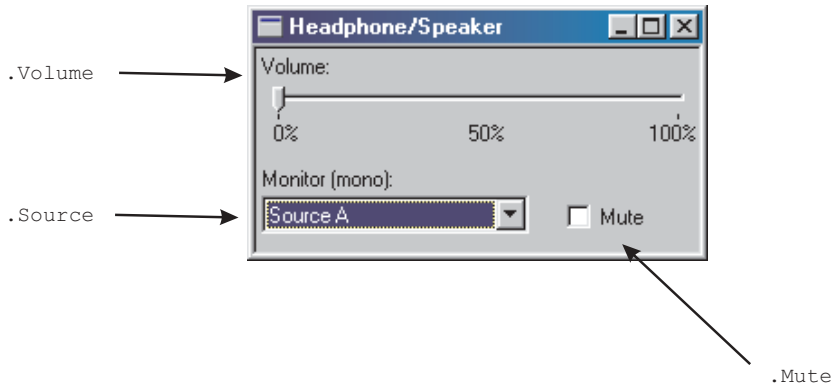


## Speaker

All commands on this page start with the following:

**ATS2.Speaker**

Example: `ATS2.Speaker.Source`

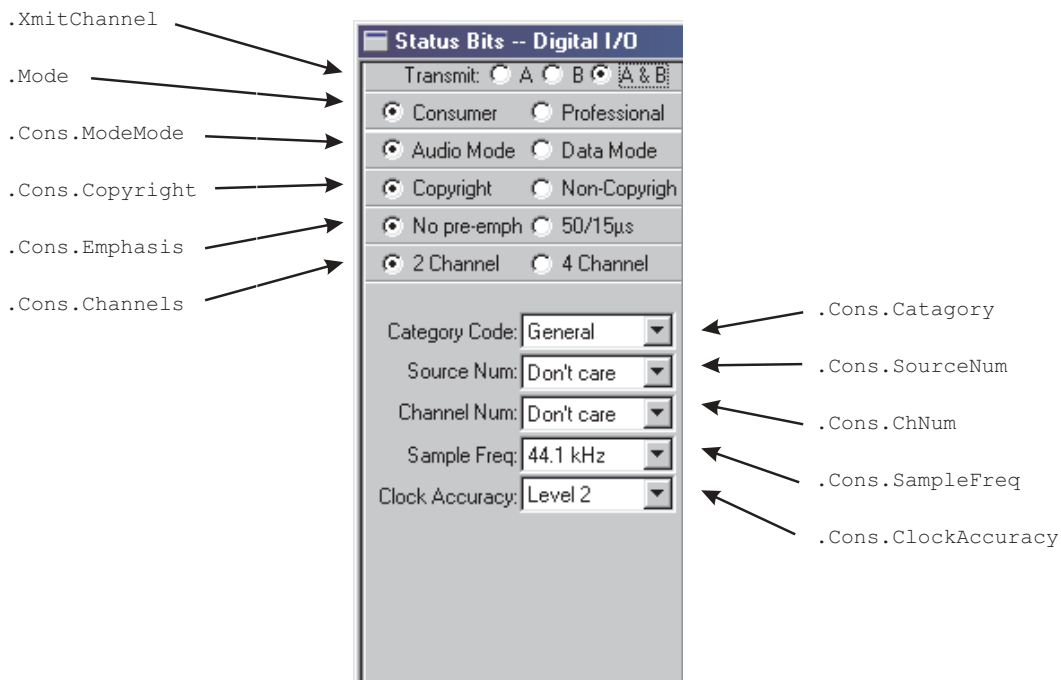


### Status Bits — Digital IO - Transmit Consumer

All commands on this page start with the following:

#### **ATS2.Bits**

Example: `ATS2.Bits.Mode`

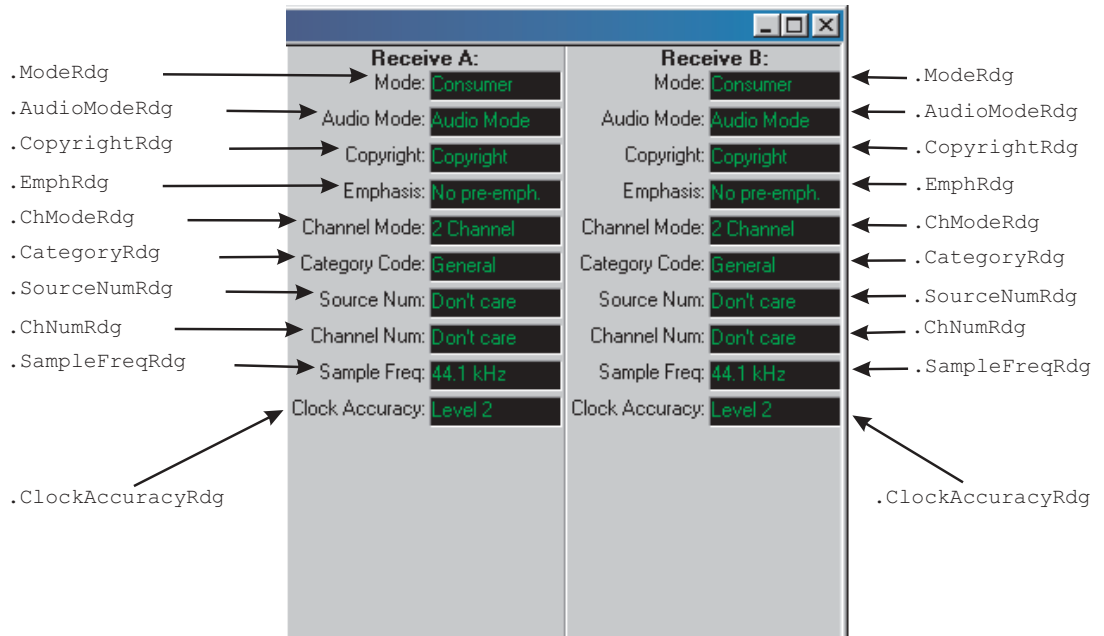


## Status Bits — Digital IO - Receive Consumer

All commands on this page start with the following:

**ATS2.Bits**

Example: ATS2.Bits.

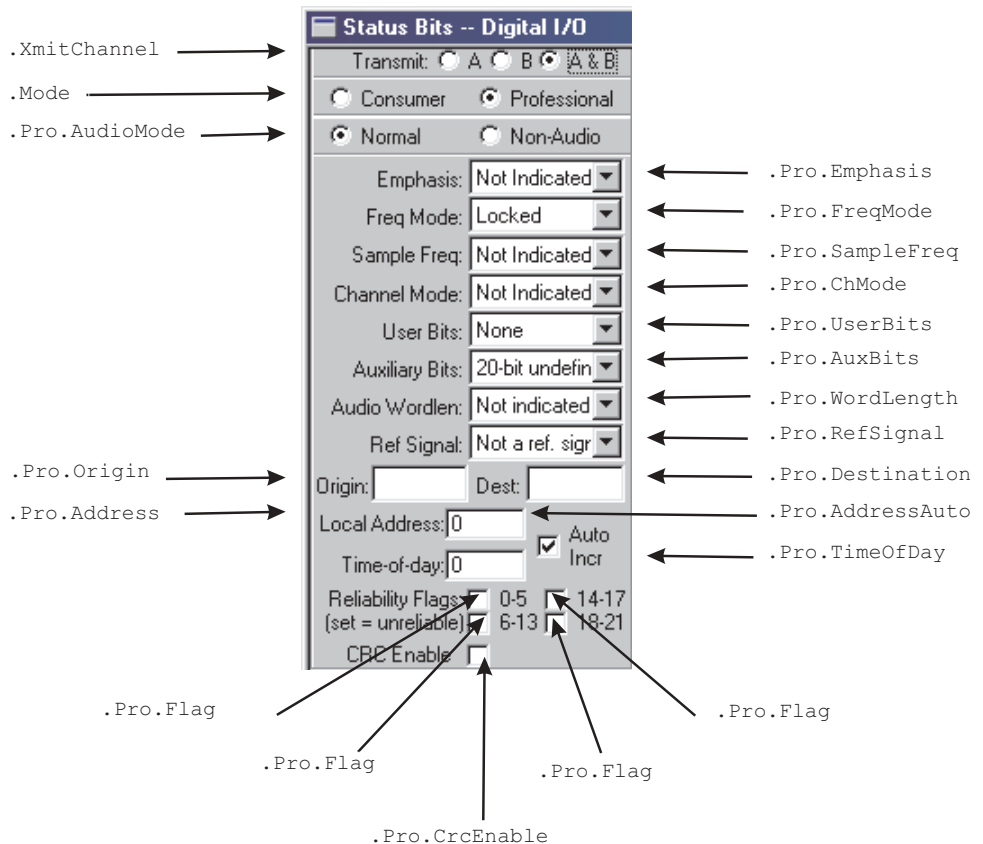


### Status Bits — Digital IO - Transmit Professional

All commands on this page start with the following:

**ATS2.Bits**

Example: `ATS2.Bits.XmitChannel`



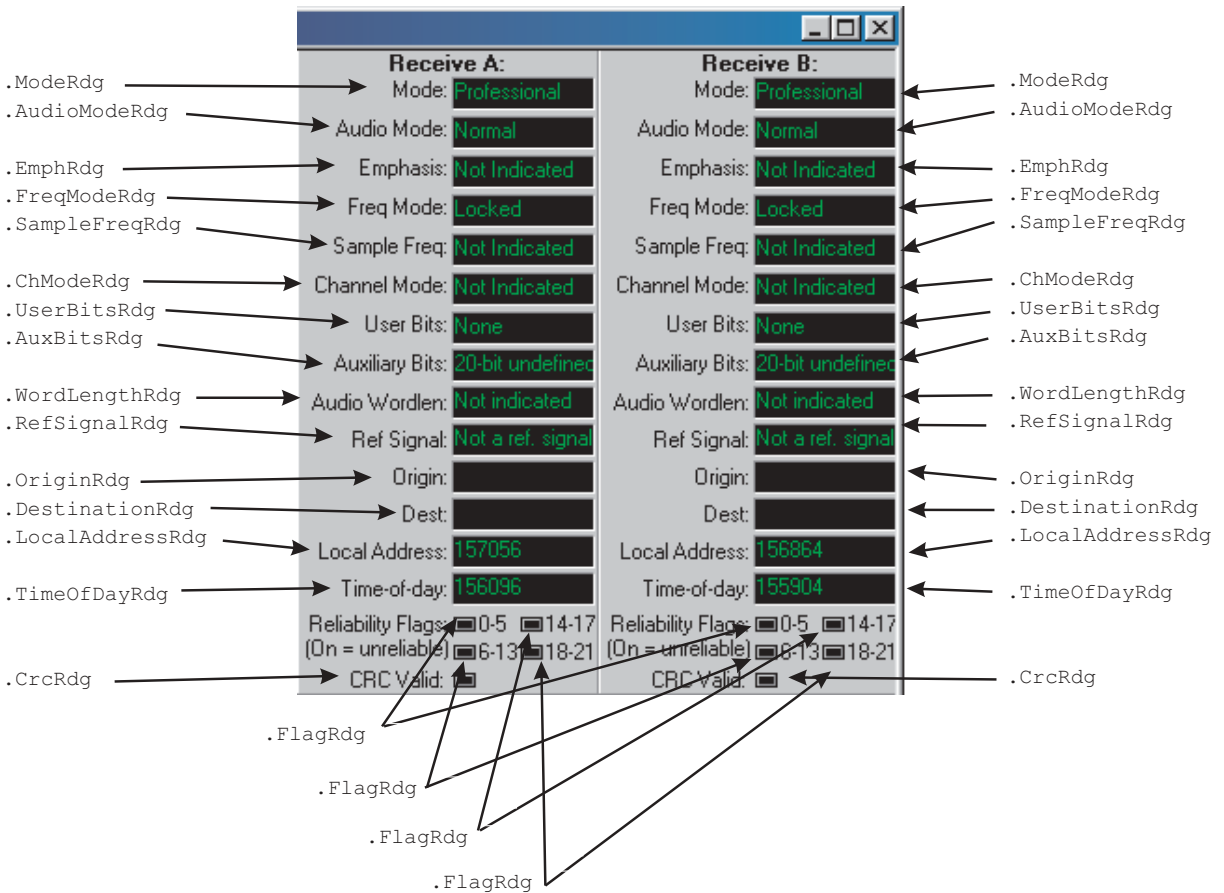
## Status Bits — Digital IO - Receive Professional

.ModeRdg

All commands on this page start with the following:

**ATS2.Bits**

Example: ATS2.Bits.



### Status Bits — Digital IO - Receive Professional

All commands on this page start with the following:

**ATS2.Bits**

Example: `ATS2.Bits.StatusXferToArray`

`.StatusXferToString`

`.XmitStatus`

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Transmit A:	01	00	00	00	00	00	80	80	80	80	80	80	80	80	00	00	00	00	00	00	00	00	00	00
Receive A:	01	00	00	00	00	00	80	80	80	80	80	80	80	80	65	02	00	C0	61	02	00	00	00	00
Transmit B:	01	00	00	00	00	00	80	80	80	80	80	80	80	80	00	00	00	00	00	00	00	00	00	00
Receive B:	01	00	00	00	00	00	80	80	80	80	80	80	80	80	C0	64	02	00	00	61	02	00	00	00

`.XmitStatus`

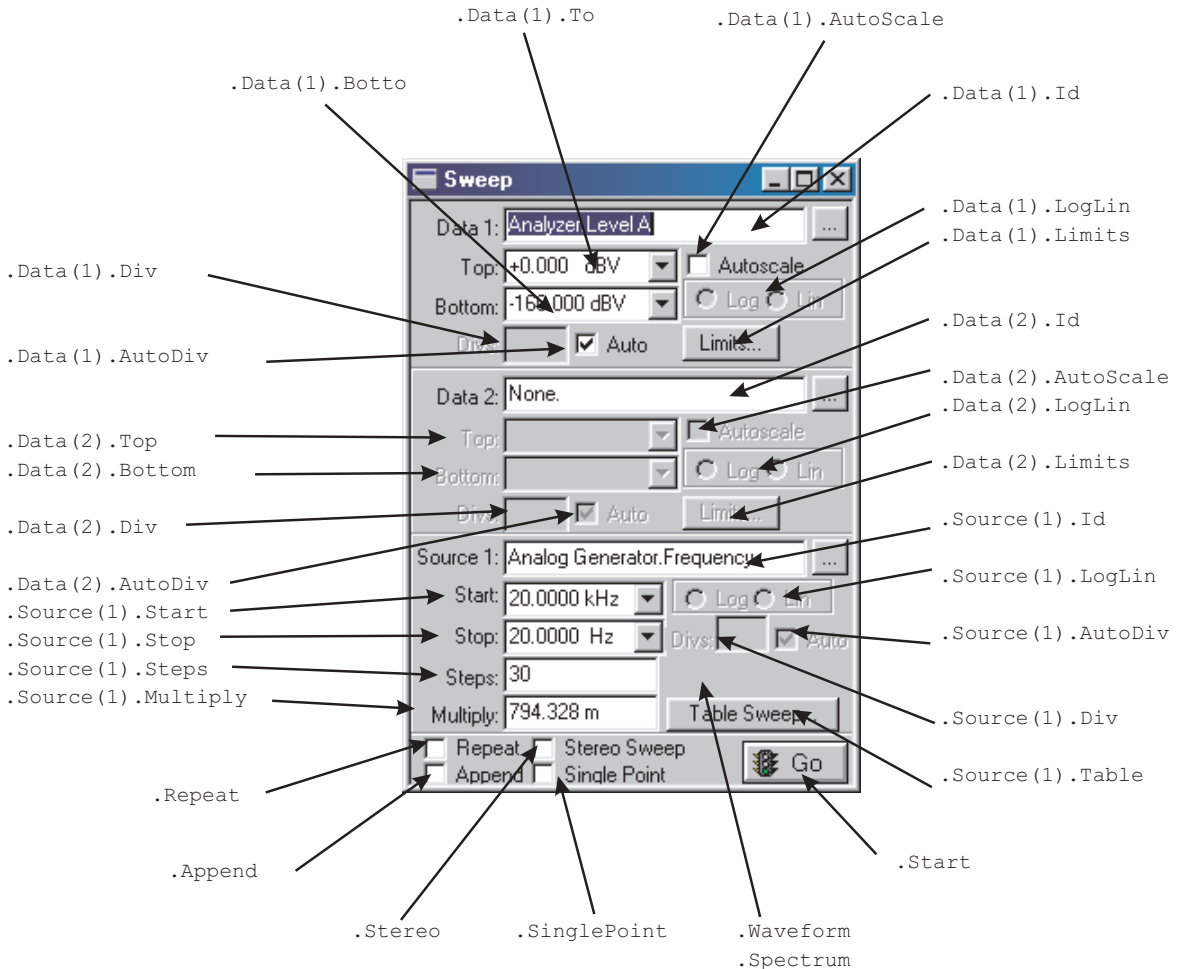
`.StatusXferToString`

## Sweep ...

All commands on this page start with the following:

### ATS.Sweep

Example: ATS.Sweep.Data(1).Id

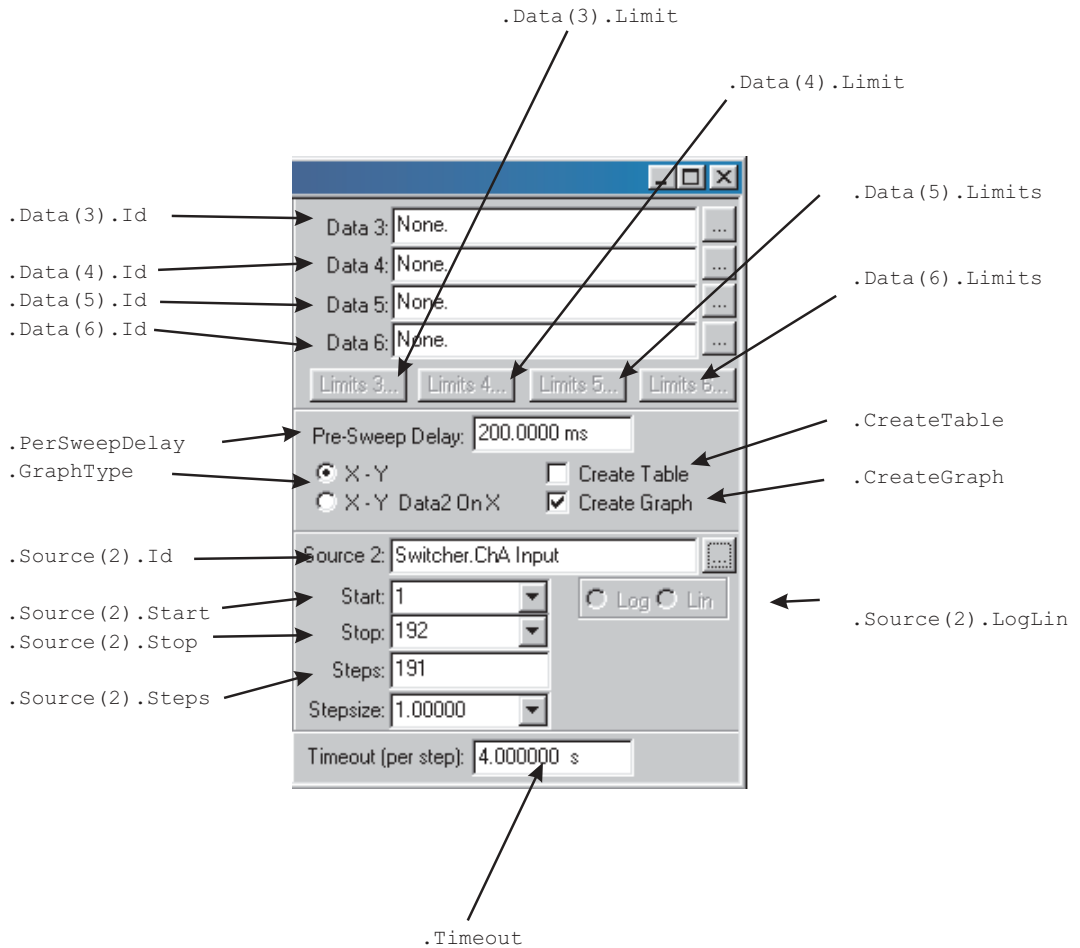


### Sweep Continued ...

All commands on this page start with the following:

**ATS.Sweep**

Example: `ATS.Sweep.Data(1).Id`



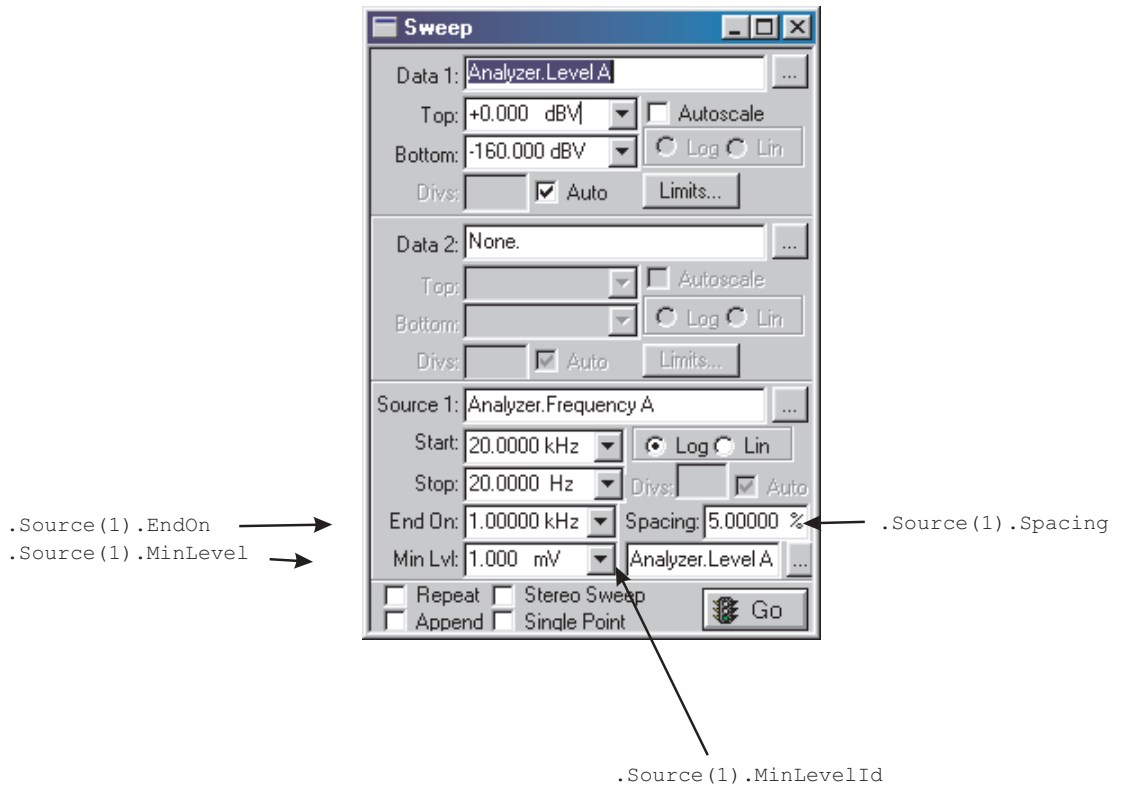


## Sweep Continued ...

All commands on this page start with the following:

### ATS.Sweep

Example: `ATS.Sweep.Source(1).EndOn`

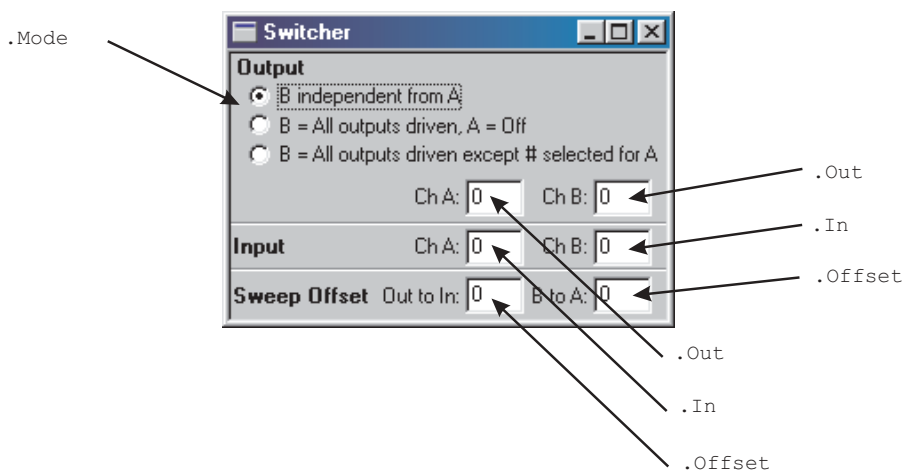


### Switcher

All commands on this page start with the following:

**ATS2.Swr**

Example: ATS2.Swr.Mode

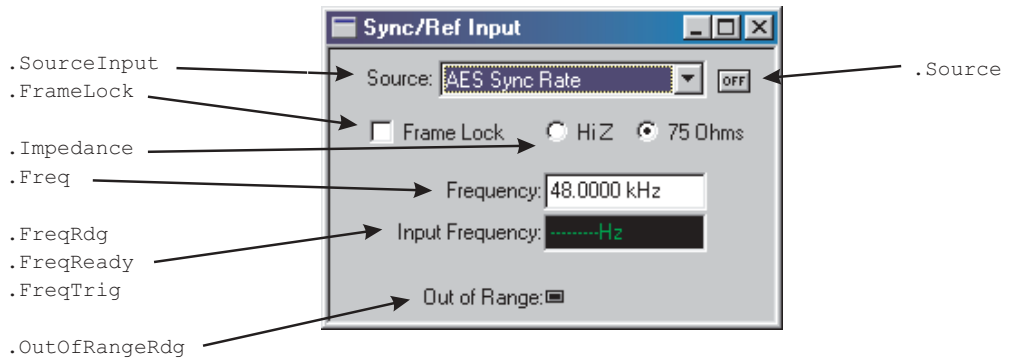


## Sync

All commands on this page start with the following:

### ATS2.Sync

Example: ATS2.Sync.Source

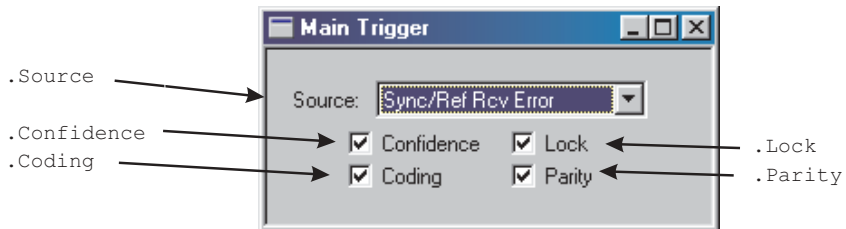


### Main Triggers

All commands on this page start with the following:

**ATS2.Sync**

Example: `ATS2.Triggers.Coding`



### ATS.Application.AppDir

Method

**Syntax**      `ATS.Application.AppDir`

**Result**        String

**Description**    This command returns the application directory. When installing the software the default application directory is "C:\Program Files\Audio Precision\ATS100\" for Windows 95, 98, ME, 2000 and NT. Utility programs are located in the application directory.

**Example**

```
Declare Function GetShortPathName Lib "kernel32" _
    Alias "GetShortPathNameA" _
    (ByVal lpLongPath As String, _
    ByVal lpShortPath As String, _
    ByVal nSizeShortPath As Long) As Long

Sub main()
    Dim LongPath As String
    Dim ShortPath As String
    ShortPath = String(255, vbNullChar)

    LongPath = ATS.Application.AppDir
    ReturnLength = GetShortPathName(LongPath, _
        ShortPath, Len(ShortPath))
    ShortPath = Left(ShortPath, ReturnLength)
    Debug.Print "Long Path = " & LongPath
    Debug.Print "Short Path = " & ShortPath
End Sub
```

**Output**

```
Long Path = C:\PROGRAM FILES\AUDIO PRECISION\ATS100\
Short Path = C:\PROGRA~1\AUDIOP~1\ATS100\
```

**ATS.Application.ClearCurrentError****Method**

**Syntax**      **ATS.Application.ClearCurrentError**

**Description**      This command when executed clears the current error.

Note: In AP Basic, "Dim WithEvents" is allowed in any module. In Visual Basic, "Dim WithEvents" is only allowed in Class modules.

See Appendix D Extensions Error Codes for Error String numbers and descriptions.

**Example**

```
Sub Main
    ATS2.AGen.Ampl (apbChA, "Vrms") = 111.9
    'Cause an error and see what happens.
End Sub

Public Sub ATSEvent_OnError (ByVal ErrorCode As Long)
    Debug.Print "Got number " & ErrorCode & " " & _
        ATS.Application.GetCurrentErrorString

    ' If you are going to handle the error, then call
    ' ATS.Application.ClearCurrentError before you exit
    ' this subroutine to stop ATS from displaying the
    ' error,

ATS.Application.ClearCurrentError

    ' It is also preferable to call
    ' ATS.Application.ClearCurrentError before you
    ' make any other calls into ATS in case these
    ' calls also generate an unexpected error
End Sub
```

**ATS.Application.CloseAll****Method**

**Syntax**      **ATS.Application.CloseAll** (ByVal *PageNum* As Integer)

Parameter	Name	Description
	<i>PageNum</i>	1 - 5 = Remove all panels from view specified page.
<b>Description</b>	This command removes all panels from view on the selected page.	
<b>Example</b>	<pre> Sub Main   ATS.Application.NewTest   ATS.Application.PanelOpen(apbAnalogGen, apbLarge)   ATS.Application.PanelOpen(apbSweep, apbSmall)   ATS2.AGen.OutputOn = True   ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon   ATS2.Inst.Analyzer.FuncFilterHP = apbAnlrHP400   ATS.Application.Page = 3   <b>ATS.Application.CloseAll</b>(2)   ATS.Application.Page = 2   ATS.Sweep.Start End Sub </pre>	

---

## ATS.Application.CloseAllPages

## Method

**Syntax**      **ATS.Application.CloseAllPages**

**Description**      This command removes all panels from view on all pages.

**Example**

```

Sub Main
  ATS.Application.NewTest
  ATS.Application.PanelOpen(apbAnalogGen, apbLarge)
  ATS.Application.PanelOpen(apbSweep, apbSmall)
  ATS2.AGen.OutputOn = True
  ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
  ATS2.Inst.Analyzer.FuncFilterHP = apbAnlrHP400
  ATS.Application.Page = 3
  ATS.Application.CloseAllPages
  ATS.Application.Page = 2
  ATS.Sweep.Start
End Sub

```

---

## ATS.Application.CopyPanelToClipboard Method

**Syntax**            **ATS.Application.CopyPanelToClipboard**

**Description**    This command copies the graphic image of the panel that has focus to the Clipboard.

**Example**

```

Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.Inst.Analyzer.FuncFilterHP = apbAnlrHP400
    ATS2.Inst.Analyzer.FuncFilterLP = apbAnlrLP20k
    ATS.Sweep.Data(1).Id = 6343
    ATS.Sweep.Source(1).Start("Hz") = 30000.0
    ATS.Sweep.Start
    ATS.Graph.OptimizeLeft
ATS.Application.CopyPanelToClipboard

    Dim MSWord As Object
    Set MSWord = CreateObject("Word.Basic") 'Start Word
    With MSWord
        .AppShow                    'Display MS Word
        .FileOpen Name:= CurDir & "\GENERIC.DOC"
        .EditFind "Place Graph Here" 'Search for string

        .EditPaste                 'Paste Graph into Word
    Wait 10
        .FileCloseAll 2            'Close all open files
        .AppClose                 'Close MS Word
    End With
End Sub

```

---

## ATS.Application.DisplayCurrentError Method

**Syntax**            **ATS.Application.DisplayCurrentError**

**Result**            None



**Description** This command temporally overrides display suppression of error messages by the error handler and displays for the current error.

**Example**

```

Sub Main
    ATS.Gen.Ampl(apbChA, "Vrms") = 111.9
    'Cause an error and see what happens.
End Sub
Public Sub ATSEvent_OnError(ByVal ErrorCode As Long)
    Debug.Print "Got number " & ErrorCode & " " & _
        ATS.Application.GetCurrentErrorString

    ' In some cases you may want the operator to see the
    ' error message. The following command will display
    ' the error.

    ATS.Application.DisplayCurrentError

    ' If you are going to handle the error, then call
    ' ATS.Application.ClearCurrentError before you exit
    ' this subroutine to stop ATS from displaying the
    ' error.

    ATS.Application.ClearCurrentError

    ' It is also preferable to call
    ' ATS.Application.ClearCurrentError before you make
    ' any other calls into ATS in case these calls also
    ' generate an unexpected error.
End Sub
    
```

---

**ATS.Application.DisplayDataOnTestOpen**

**Property**

**Syntax**            **ATS.Application.DisplayDataOnTestOpen**

**Data Type**        Boolean

*True*                Display data on test open.

*False*               Don't display data on test open.

**Description** This command specifies whether the measurement data saved in a test file is displayed when the file is loaded.

**Example**

```
Sub Main
    ATS.Application.DisplayDataOnTestOpen = False
    ATS.File.OpenTest "SAMPLE1.ATS2"
    String1$ = "Test Loaded and data NOT displayed."
    ATS.Prompt.Text = String1$
    ATS.Prompt.FontSize = 10
    ATS.Prompt.Position -1,-1,290,120
    ATS.Prompt.ShowWithContinue
    Stop

    ATS.Application.DisplayDataOnTestOpen = True
    ATS.File.OpenTest "SAMPLE1.ATS2"
    String1$ = "Test loaded and data displayed."
    ATS.Prompt.Text = String1$
    ATS.Prompt.FontSize = 10
    ATS.Prompt.Position -1,-1,290,100
    ATS.Prompt.ShowWithContinue
    Stop
End Sub
```

---

**ATS.Application.DoReadings****Method**

**Syntax** `ATS.Application.DoReadings`

**Result** None

**Description** This command forces a reading cycle to take place. The reading cycle allows the reading commands (commands ending in Rdg such as `ATS2.Inst.Analyzer.FuncRdg`) to make and return a measurement. Under normal conditions when a dialog is displayed the automatic readings cycle is disabled and readings will not return correctly. Use this command to force a reading cycle to take place while a dialog is displayed.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
```

```

ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
ATS2.AGen.Ampl(apbChA, "Vrms") = 0.5

Begin Dialog UserDialog 270,105,"Readings _
    Example",.Handler 'GRID:10,7,1,1
    PushButton 40,7,190,21,"Make _
        Reading",.PushButton1
    TextBox 40,35,190,21,.TextBox1
    CancelButton 40,63,190,21
End Dialog
Dim dlg As UserDialog
Select Case Dialog (dlg)
Case 0
    End
End Select
End Sub

Private Function Handler(DlgItem$, Action%, _
    SuppValue&) As Boolean
    Select Case Action%
    Case 1
    Case 2
        If DlgItem$ = "PushButton1" Then
            Handler = True
            ATS2.Inst.Analyzer.FuncSettling(apbChA, 5.0,
                1.0e-007, "V", 3, 0.05, apbExponential)
            ATS2.Inst.Analyzer.FuncTrig(apbChA)
            Do
                ATS.Application.DoReadings
            Loop Until ATS.Analyzer.FuncReady(apbChA)
            AReading = ATS2.Inst.Analyzer.FuncRdg _
                (apbChA, "V")
            DlgText "TextBox1", "Amplitude = " & _
                Format(AReading, "#.00000") & " V"
        End If
    Case 3
        'TextBox or ComboBox text changed
    Case 4
        'Focus changed
    Case 5
        'Idle
        Rem Handler = True 'Continue getting idle
actions
    Case 6
        'Function key

```

```

    End Select
End Function

```

---

## ATS.Application.GetCurrentErrorString

**Method**

- Syntax**      **ATS.Application.GetCurrentErrorString**
- Result**        String
- Description**    This command returns the ASCII text string for the current error.  
See Appendix D Extensions Error Codes for Error String numbers and descriptions.
- Example**        See example for `ATS.Application.ClearCurrentError`.

---

## ATS.Application.GetEnumString

**Method**

- Syntax**        **ATS.Application.GetEnumString**(ByVal *Function* As String, ByVal *Enumeration* As Integer)
- Result**        String
- | Parameter | Name               | Description            |
|-----------|--------------------|------------------------|
|           | <i>Function</i>    | OLE command extension. |
|           | <i>Enumeration</i> | Enumerated Constant.   |
- Description**    This command returns the ASCII text string for the specified Enumerated Constant.
- Example**        Sub Main  
                   ATS2.AnalogIn.Source(apbChA) = apbAnalogInBNC\_Unbal  
                   Debug.Print apbAnalogInBNC\_Unbal  
                   Debug.Print **ATS.Application.GetEnumString** \_  
                   ("ATS2.AnalogIn.Source", apbAnalogInBNC\_Unbal)  
                   End Sub
- Output**        1847  
                   apbAnalogInBNC\_Unbal

## ATS.Application.Input

## Method

**Syntax** `ATS.Application.Input (ByVal PortAddress As Integer)`

Parameter	Name	Description
	<i>PortAddress</i>	An I/O address value between 0 and 65535 (FFFF Hex).

**Result** Integer

**Description** The purpose of this command is to read input from an external device through a parallel port, or an I/O mapped interface card plugged into the computer.

The decimal read address of the first printer port (treated by DOS as LPT1) is 889 (379 Hex) and the second port (LPT2) is 633 (279 Hex).

The standard parallel port has four pins that can be used for Input.; pins 11, 12, 13, and 15. Each line is held high by an internal pull-up resistor and requires approximately 1mA to pull the line low, this will allow other parallel ports to drive the input.

When all of the input lines (11(), 12(), 13(), and 15()) are high the `ATS.Application.Input` command will return decimal 127. The following list shows the return value for each line when it is pulled low.

Note: This command is not available for the Windows 2000 or NT operating system.

Input Line	Bit Value		
Data Line	Return Value		
11	128	7	255
12	32	5	95
13	16	4	111
15	8	3	119

**Example**

```

Sub Main
Return = ATS.Application.Input(889)
If Return = 127 Then Debug.Print "All Lines _
    pulled high."
If Return And 128 Then      'Using And logic.
    
```

```

        Debug.Print "Pin 11 pulled low."
    End If
    If Return Xor 32 Then      'Using Xor logic.
        Debug.Print "Pin 12 pulled low."
    End If
    If Not Return And 16 Then 'Using Not And logic.
        Debug.Print "Pin 13 pulled low."
    End If
    If Not Return And 8 Then  'Using Not And logic.
        Debug.Print "Pin 15 pulled low."
    End If
End Sub

```

**Output** All Lines pulled high.

---

## ATS.Application.MacroDir

(OLE) Method

**Syntax**      **ATS.Application.MacroDir**

**Result**      String

**Description**      This command returns the running macro source directory. This command is like the MacroDir\$ command in the Language reference section of AP Basic with the exception that this command can be used from an OLE client that is accessing ATS to determine the directory from which the selected macro was loaded.

**Example**

```

Private Sub Form_Load()
    Dim ATS As Object
    Dim ATS2 As Object
    Set ATS = CreateObject("ATS.Application")
    Set ATS2 = CreateObject("ATS.ATS-2")

```

```

'The following lines makes the Visual Basic Current
' Directory and the ATS Working Directory the same
' as the directory where the current AP Basic macro
' was loaded from.

```

```

With ATS.Application

```

```

        .Visible = True
    ATS.File.OpenMacro ("C:\BUSY.ATSB")
    ChDir .MacroDir
    .WorkingDir = .MacroDir
End With

'Your code goes here.

End Sub

```

## ATS.Application.Name

## Method

**Syntax**            **ATS.Application.Name**

**Result**            String            ASCII characters.

**Description**      This command returns the ATS Application Name “Audio Precision ATS”. This text string is located in the upper left corner of the ATS application before the test name. This string is useful when using the AppActivate command located in the Language reference section of AP Basic.

**Example**

```

Sub Main
    AppActivate ATS.Application.Name
    SendKeys "%WC",1            'Clear all windows on page.
    SendKeys "%PO",1            'Display Data Editor.

    'In Debug mode focus is automatically returned to
    ' the editor each time the user interacts with the
    ' controls. Therefore it is important to note that
    ' sections of code containing commands that are to
    ' be sent to other applications via the SendKeys
    ' command need to be executed without interruption.

    'When debugging these areas place a breakpoints
    ' before and after the SendKeys commands to
    ' maintain
    ' the correct window/application focus.
End Sub

```

---

**ATS.Application.NewData****Method****Syntax**      **ATS.Application.NewData****Result**      Boolean

*True*              Data removed from memory.  
*False*             Command failed to remove data from memory.

**Description**      This command deletes the measurements currently in memory. The command is functionally the same as selecting File, New, Data from the Menu bar.**Example**

```
Sub Main
  ATS.File.OpenTest "FRQ-RESP.ATS2"
  ATS.Application.NewData
  ATS.Sweep.Start
  ATS.File.SaveDataAs "FRQ-RESP.ATSA"
  ATS.File.OpenTest "THD-FRQ.ATS2"
  ATS.Application.NewData
  ATS.Sweep.Start
  ATS.File.SaveDataAs "THD-FRQ.ATSA"
  ATS.File.OpenTest "RESIDNOI.ATS2"
  ATS.Application.NewData
  ATS.Sweep.Start
  ATS.File.SaveDataAs "RESIDNOI.ATSA"
End Sub
```

---

**ATS.Application.NewMacro****(OLE) Method****Syntax**      **ATS.Application.NewMacro****Result**      Boolean

*True*              New macro created.  
*False*             Command failed to create new macro.

**Description**      This command initializes the macro editor and is only to be used via OLE. The command is functionally the same as selecting File, New, Macro, and OK from the Menu bar.



## ATS.Application.NewTest

## Method

**Syntax**            **ATS.Application.NewTest**

**Result**            Boolean

*True*                New test panel configuration restored.  
*False*                Command failed to restore new test panel configuration.

**Description**    This command initializes the current ATS test to the default test condition. The command is functionally the same as selecting New Test from the Standard Toolbar or selecting File, New, Test, and OK from the Menu bar.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.Inst.Analyzer.FuncFilterLP = apbAnlrLP20k
    A = ATS2.Inst.Analyzer.FuncFilterLP
    ATS.Sweep.Source(1).Start("Hz") = 30000
    ATS.Sweep.Start
End Sub
```

## ATS.Application.Output

## Method

**Syntax**            **ATS.Application.Output** (ByVal *PortAddress* As Integer, ByVal *DataByte* As Integer)

**Parameter**

Name	Description
<i>PortAddress</i>	An I/O address value between 0 and 65535 (FFFF Hex).
<i>DataByte</i>	Any value between 0 and 255(FF Hex).

**Description**    The purpose of this command is to control an external device through a parallel port, or an I/O mapped interface card plugged into the computer.

The standard parallel port has eight pins that can be used for Output.; pins 2, 3, 4, 5, 6, 7, 8, and 9. The decimal address of the first printer port (treated by DOS as LPT1) is 888 (378 Hex) and the second port (LPT2) is 632 (278 Hex).

Note: This command is not available for the Windows 2000 or NT operating system.

**Example**

```
Sub Main
  'Set all LPT1 bits high.
  ATS.Application.Output(888, 255)
End Sub
```

**ATS.Application.Page****Property**

**Syntax**            **ATS.Application.Page**

**Data Type**        Integer

1	Page #1.
2	Page #2.
3	Page #3.
4	Page #4.
5	Page #5.

**Description**     This command displays the selected page,

**Example**

```
Sub Main
  ATS.Application.NewTest
  ATS.Application.Page = 1
  Return = ATS.Application.Page
  Debug.Print "Page "; Return; " displayed."
  Wait 1 'So the user can see the page change.
  ATS.Application.Page = 2
  Return = ATS.Application.Page
  Debug.Print "Page "; Return; " displayed."
  Wait 1
  ATS.Application.Page = 3
  Return = ATS.Application.Page
  Debug.Print "Page "; Return; " displayed."
End Sub
```

**ATS.Application.PanelClose****Method**

**Syntax**      **ATS.Application.PanelClose** (ByVal *PanelID* As Constant)

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>PanelID</i>	apbAnalogGen = Remove Analog Generator panel from view. apbAnalyzer = Remove the Analyzer panel from view. apbBarGraph? = Remove the desired Bar Graph 1 through 32 from view. apbDataEditor = Remove the Data Editor panel from view. apbDCX = Remove the DCX-127 panel from view. apbDiagnostic = Remove the Diagnostic panel from view. apbDigIO = Remove the Digital Input / Output panel from view. apbDigitalGen = Remove the Digital Generator panel from view. apbDIOStatusBits = Remove the Status Bits panel from view. apbGraph = Remove the Graph from view. apbRegulatio = Remove the Regulation panel from view. apbSettling = Remove the Settling panel from view. apbSpeaker = Remove the Speaker panel from view. apbSweep = Remove the Sweep panel from view. apbSwitcher = Remove the Switcher panel from view.

**Description**      This command closes the selected panel,

**Example**

```

Sub Main
    ATS.Application.NewTest
    ATS.Application.PanelOpen(apbSweep, apbSmall)
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.Inst.Analyzer.FuncFilterHP = apbAnlrHP400
    ATS.Application.Page = 3
    ATS.Application.PanelClose(apbDigIO)
    ATS.Application.Page = 2
    ATS.Sweep.Start
End Sub

```

**ATS.Application.PanelOpen****Method**

**Syntax**      **ATS.Application.PanelOpen** (ByVal *Panel* As Constant[, ByVal *Size* As Constant])

Parameter	Name	Description
	<i>Panel</i>	apbAnalogGen = Display Analog Generator panel. apbAnalyzer = Display Analyzer panel. apbBarGraph? = Display Bar Graph panel 1 through 32. apbDataEditor = Display Data Editor panel. apbDCX = Display DCX-127 panel. apbDiagnostic = Display Diagnostic panel. apbDigIO = Display Digital Input / Output panel. apbDigitalGen = Display Digital Generator panel. apbDIOStatusBits = Display Status Bits panel. apbGraph = Display Graph panel. apbRegulation = Display Regulation panel. apbSettling = Display Settling panel. apbSpeaker = Display Speaker panel. apbSweep = Display Sweep panel. apbSwitcher = Display Switcher panel.
	<i>Size</i>	apbSmall = Display small version of panel apbLarge = Display large version of panel

**Description**      This command displays the selected panel on the current page.

**Example**              See example for `ATS.Application.PanelClose`.

**ATS.Application.Quit****Method**

**Syntax**              **ATS.Application.Quit**

**Description**      This command terminates ATS and returns to Windows. If the "Prompt to Save Test when a test is closed" selection in the Utilities, Config menu is enabled the operator will be prompted to save changed files when ATS-2 quits.

**Example**              Sub Main  
                             Start:

```

ChDir MacroDir
Begin Dialog UserDialog 430,105
    PushButton 20,21,380,28,"Your Code",.Field1
    PushButton 130,63,180,28,"Exit ATS",.Field3
End Dialog
Dim Main_Menu As UserDialog
Select Case Dialog(Main_Menu)
    Case 1
        'Incert your code here...
    Case Else
        ATS.Application.Quit 'Exit ATS
End Select
GoTo Start:
End Sub

```

---

## ATS.Application.Restore

## Method

**Syntax**      **ATS.Application.Restore**

**Description**      This command restores the hardware to the present state of the software.

This function should be used if the hardware loses power or becomes disconnected from the computer.

**Example**

```

Sub Main
    Start:
    Begin Dialog UserDialog 430,105,"Example Menu"
        PushButton 40,28,170,42,"Restore _
        Hardware",.Field1
        PushButton 230,28,160,42,"Exit Macro",.Field2
    End Dialog
    Dim Main_Menu As UserDialog

    Select Case Dialog(Main_Menu)
        Case 1
            ATS.Application.Restore
        Case Else
            End
    End Select
End Sub

```

```

    GoTo Start:
End Sub

```

## ATS.Application.SetWatchDogTimer

## Method

**Syntax** **ATS.Application.SetWatchDogTimer** (ByVal *Number* As Integer, ByVal *Sec* As Double [, Optional ByVal *ThrowError* As Variant])

Parameter	Name	Description
	<i>Number</i>	1 or 2 Two timers can be defined.
	<i>Sec</i>	Defines the amount of time that will elapse after starting the WatchDog Timer before the ATSEvent_OnWatchDogTimeout event is generated. To disable the timer at any time set the time value to (0) zero seconds.
	<i>ThrowError</i>	Optional parameter. Set this parameter to True to throw/raise an error (11021) when the defined time has elapsed. Basic's On Error mechanism can then detect the error. The default (False) condition will not throw/raise an error when the defined time has elapsed.

**Description** This command sets up and starts the defined timer. When the defined time expires the ATSEvent\_OnWatchDogTimeout event is generated. In addition an error can be thrown/raised to allow Basic's On Error mechanism to intercept the error.

**Example**

```

Dim Halt As Boolean
Sub Main
    Halt = False
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source (apbChA) = apbAnalogInGenMon
    ATS.Sweep.Source(1).Steps = 30
    ATS.Application.SetWatchDogTimer(1, 5.0 ,False)
    ATS.Sweep.StartNoWait
    Do
        Loop While Halt = False
End Sub

```

```

Sub ATSEvent_OnWatchDogTimeout(ByVal Id As Long)
    If Id = 1 Then
        If ATS.Sweep.IsRunning = True Then
            ATS.Sweep.Stop
            Debug.Print "Sweep Stopped"
        End If
    End If
End Sub

```

---

## ATS.Application.SysType

## Method

**Syntax**      **ATS.Application.SysType**

**Result**      String

**Description**      This command returns the path of the test (.AT2R) that is currently loaded.

**Example**

```

Sub Main
    Select Case (ATS.Application.SysType)
        Case "ATS-2"
            ATS.Prompt.Text = "ATS configured for _
                ATS-2 hardware."
        End Select
    ATS.Prompt.ShowWithContinue
    Stop
End Sub

```

---

## ATS.Application.TestDir

## Method

**Syntax**      **ATS.Application.TestDir**

**Result**      String

**Description**      This command returns the path of the test (.ATS2) that is currently loaded.

**Example**

```

Sub Main
    ATS.Application.DisplayDataOnTestOpen = True

```

```

ATS.File.OpenTest "SAMPLE1.ATS2"
TestName$ = ATS.Application.TestName
TestDir$ = ATS.Application.TestDir
String1$ = "Test file "
String2$ = " was loaded from the "
String3$ = " directory."
ATS.Prompt.Text = String1$ & TestName$ & String2$ _
    & TestDir$ & String3$
ATS.Prompt.FontSize = 10
ATS.Prompt.Position -1,-1,300,150
ATS.Prompt.ShowWithContinue
Stop
End Sub

```

---

## ATS.Application.TestName

## Method

<b>Syntax</b>	<b>ATS.Application.TestName</b>
<b>Result</b>	String
<b>Description</b>	This command returns the test (.ATS2) file name of the test that is currently loaded.
<b>Example</b>	See example for <code>ATS.Application.TestDir</code> .

---

## ATS.Application.ThrowErrors

## Property

<b>Syntax</b>	<b>ATS.Application.ThrowErrors</b>
<b>Data Type</b>	Boolean
	<i>True</i> Expose Errors and Warnings.
	<i>False</i> Don't expose Errors and Warnings..
<b>Description</b>	This command exposes errors and warnings generated by ATS to the Err. object.
	See Appendix D Extensions Error Codes for Error String numbers and descriptions.
<b>Example</b>	Sub Main



```

' Pick one the three On Error possibilities below
'On Error GoTo 0 'Disable your error handler
                '(default). Basic will handle
                'the error by termination.
On Error GoTo MyErrorHandler ' Send error
                'conditions to "MyHandler"
'On Error Resume Next ' Error conditions continue
                'execution at the next statement.
'Caution - This is generally very dangerous
' as no error will be seen
'ATS will not send errors to Basic's
' "On Error" unless ATS.Application.
' ThrowErrors is set to True
'After this is set to True, ATS will
' no longer display errors, they will all
' be passed to Basic
ATS.Application.ThrowErrors = True
'Now cause an error and see what happens.
ATS2.AGen.Ampl(apbChA, "Vrms") = 111.9
' Cause another error and see what happens.
ATS2.AGen.Freq(apbChA, "Hz") = 1.0
MsgBox "Resumed after the offending Call"
Exit Sub' Exit to avoid handler.

```

MyErrorHandler:

```

'Show some debug info
Debug.Print "Err=";Err.Number
Debug.Print "Description=";Err.Description
Debug.Print "Source=";Err.Source

'Select different actions for errors
Select Case Err.Number ' Evaluate error number
Case 8504
    '"Generator Amplitude" error.
    'Put your error handler code here
    MsgBox "Got to the Handler"
    'If you handled the error, then resume
    Resume Next
Case Else
    'Handle other situations here...

```

```

'If we don't know about the err then or
' instead you could decide that Basic
' Should handle this
'Note that if this is a called subroutine,
' Basic will pass the error back to the
' calling subroutine.
'Passing all the info:
Err.Raise(Err.Number, Err.Source _
        ,Err.Description)
' Or more simply
Error Err
' Or you could exit this subroutine
Exit Sub

End Select
End Sub

```

---

## ATS.Application.Version

## Method

**Syntax**      **ATS.Application.VisibleMacroEditor**

**Result**      Double

**Description**      This command returns the running ATS Application Version number. This command can be used to check if the running version of ATS is compatible with the running macro.

**Example**

```

Sub Main
  If ATS.Application.Version < 1.0 Then End
  ATS.Application.NewTest
  ATS.Application.PanelOpen(apbSweep, apbSmall)
  ATS2.AGen.OutputOn = True
  ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
  ATS2.Inst.Analyzer.FuncFilterHP = apbAnlrHP400
  ATS.Application.Page = 3
  ATS.Application.PanelClose(apbDigIO)
  ATS.Application.Page = 2
  ATS.Sweep.Start
End Sub

```

---

**ATS.Application.Visible****Property****Syntax**      **ATS.Application.Visible****Data Type**    Boolean*True*            Restore ATS to view.*False*           Remove ATS from view.**Description**    This command when executed makes the ATS window visible or invisible. The Macro Editor remains visible.**Example**            Sub Main  
                          **ATS.Application.Visible** = False  
                          ATS.Application.NewTest  
                          ATS2.AGen.OutputOn = True  
                          ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon  
                          ATS.Sweep.Start  
                          **ATS.Application.Visible** = True  
                          End Sub

---

**ATS.Application.VisibleMacroEditor****Property****Syntax**            **ATS.Application.VisibleMacroEditor** (ByVal  
                          *bVisible* As Boolean)**Data Type**        Boolean*True*            Restore Macro Editor to view.*False*           Remove Macro Editor from view.**Description**        This command when executed makes the ATS Macro Editor visible or invisible. Dialogs displayed when the Macro Editor is invisible have a higher Z-order (which window is on top of another) than the ATS window, therefore when focus is moved to the ATS window the dialog remains displayed on top of the ATS application. If the Macro Editor is visible then the Z-order is relative to the Macro Editor and the dialog may be covered by any other window that has focus.**Example**            Sub Main  
                          **ATS.Application.VisibleMacroEditor** = False

```

ATS.Application.NewTest
ATS2.AGen.OutputOn = True
ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
ATS.Sweep.Start
ATS.Application.VisibleMacroEditor = True
End Sub

```

## ATS.Application.VisibleOnTestLoad

## Property

**Syntax**      **ATS.Application.Visible** (ByVal ID As Constant)

**Data Type**      Boolean

*True*              Restore to view.  
*False*             Remove from view.

**Parameter**

Name	Description
<i>ID</i>	<p>apbAll = This constant enables or disables display of the Graph and Bar-Graph displays, Data Editor, and Panels when a test is loaded during Macro execution only.</p> <p>apbBarGraphs = This constant enables or disables display of the Bar-Graph display, when a test is loaded during Macro execution only. Use this command at the beginning of your macro to decrease overall test times.</p> <p>apbDataEditor = This constant enables or disables display of the Data Editor panel when a test is loaded during Macro execution only. Use this constant at the beginning of your macro to decrease overall test times.</p> <p>apbGraph = This constant enables or disables display of the Graph display when a test is loaded during Macro execution only. Use this constant at the beginning of your macro to decrease overall test times.</p> <p>apbPanels = This constant enables or disables display of the Panels when a test is loaded during Macro execution only. Use this constant at the beginning of your macro to decrease overall test times.</p>

**Description**      This command enables or disables display of specified sections when a test is loaded during Macro execution only. Use this command at the beginning of your macro to decrease overall test times.

**Example**

```

Sub Main
    ATS.Application.VisibleOnTestLoad(apbAll) = False
    ATS.File.OpenTest ("SAMPLE1.ATS2")
    Wait 5
    ATS.Application.VisibleOnTestLoad(apbPanels) = True
    ATS.File.OpenTest ("SAMPLE1.ATS2")
    Wait 5
    ATS.Application.VisibleOnTestLoad(apbGraphs) = True
    ATS.File.OpenTest ("SAMPLE1.ATS2")
End Sub

```

**ATS.Application.WorkingDir****Property****Syntax**

```
ATS.Application.WorkingDir
```

**Data Type**

```
String
```

**Description**

This command sets or returns the current working directory. This command is like the `ChDir$` command in the Language reference section of AP Basic with the exception that this command can be used from an OLE client to change the ATS working directory.

**Example**

```

Private Sub Form_Load()
    Dim ATS As Object
    Set ATS = ("ATS.Application")
    'The following line makes the ATS Working Directory
    ' the same as the VB current directory.
    If ATS.Application.AppDir <> CurDir Then
        ATS.Application.WorkingDir = CurDir
        'Your code goes here.
    End If
End Sub

```

## User Notes

### ATS.Aux.ReadingRdg

Property

**Syntax** `ATS.Aux.ReadingRdg (ByVal Number As Constant)`

**Data Type** Double

Parameter	Name	Description
	<i>Number</i>	apbAux1 = Auxiliary Instrument 1 apbAux2 = Auxiliary Instrument 2 apbAux3 = Auxiliary Instrument 3 apbAux4 = Auxiliary Instrument 4

**Description** This command returns a settled reading for Auxiliary Instrument Reading and zeros the ready count.

**See Also** `ATS.Aux.ReadingReady`, `ATS.Aux.ReadingSettling`, `ATS.Aux.ReadingTrig`

**Example**

```
' Uses the ATS-GPIB Library GPIBLIB.ATSB
' A National Instruments GPIB card must be installed
' in your system to use this file.
'#Uses "gpib-lib.atsb"
' See the GPIB-LIB.atsb file for instructions on use.
'#Uses "APNiglob.bas"
'#Uses "APVbib32.bas"
Public iAP As Integer

Sub Main
Dim iStatus As Integer, iAddr As Integer, _
    iAnyoneHome As Integer, sResponse As String

    iAP = ildev(0, 1, NO_SAD, T3s, 1, 0) 'Open I/O to _
        this GPIB address ** Assumes Board 0, Address 1
```

```

iStatus = illn(iAP, 1, NO_SAD, iAnyOneHome)
'Check for listener at address 1
If iAnyOneHome Then 'Found a listener at address 1
    iStatus = ilclr(iAP) 'Device clear
    GpibWrite iAP, "*IDN?" 'Query for Instrument _
        Identificaiton
    sResponse = GpibRead(iAP, 60) 'Get response
    If Not (ibsta And EERR) Then 'If no GPIB _
        read error Then save response string
        GpibWrite iAP, "*CLS;*RST;"
'Clear status registers and reset all settings
    End If
End If

result = GpibWrite(iAP, ":HEADER OFF;")
result = GpibWrite(iAP, ":SETTLE OFF;")
result = GpibWrite(iAP, ":OUTPUT ON;")

ATS.File.OpenTest("AUX GPIB Example.at2R")
ATS.Aux.ReadingSettling(apbAux1, 1.0, 0.0000001, 3,
0.03, apbExponential)
ATS.Aux.Setting(apbAux1) = 1000.0
'Set frequency for sweep to return to when done
    ATS.Sweep.Start
End Sub
Sub ATSEvent_OnSweepTrigger()
ATS.Aux.ReadingTrig(apbAux1)
While ATS.Aux.ReadingReady(apbAux1) = 0
    result = GpibWrite(iAP, ":M1?;")
    Str1 = GpibRead(iAP,80)'Read METER M1 result

    Debug.Print "Reading = " & Str1
ATS.Aux.SetReading(apbAux1, Val(Str1))
    Wait .1
Wend
    Debug.Print "Settled Reading = " & Str1
End Sub
Sub ATSEvent_OnSweepStep(Value As Variant, Source As
-
    Long)
    Value = Format(Value, "###.####")

```



```

Debug.Print "Setting = " & Value
' GPIB Code to set ATS-2 Generator Frequency
' to "Value" variable
result = GpibWrite(iAP, ":GFREQUENCY " & _
    Str$(Value)&"")      'Set Aux Generator Freq
End Sub
    
```

## ATS.Aux.ReadingReady

## Property

**Syntax**      **ATS.Aux.ReadingReady** (ByVal *Number* As Constant)

**Data Type**      Integer

0                      Reading not ready.  
 >0                     Reading ready.

Parameter	Name	Description
	<i>Number</i>	apbAux1 = Auxiliary Instrument 1 apbAux2 = Auxiliary Instrument 2 apbAux3 = Auxiliary Instrument 3 apbAux4 = Auxiliary Instrument 4

**Description**      This command returns the Auxiliary Instrument Reading settled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the `ATS.Aux.ReadingRdg` or `ATS.Aux.ReadingTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS.Aux.ReadingRdg` command will be guaranteed to return quickly.

**See Also**            `ATS.Aux.ReadingRdg`, `ATS.Aux.ReadingSettling`,  
`ATS.Aux.ReadingTrig`

**Example**             See example for `ATS.Aux.ReadingRdg`.

**ATS.Aux.ReadingSettling****Method**

<b>Syntax</b>	<b>ATS.Aux.ReadingSettling</b> (ByVal <i>Number</i> As Constant, ByVal <i>Tolerance</i> As Double, ByVal <i>Floor</i> As Double, ByVal <i>FloorUnit</i> As String, ByVal <i>Points</i> As Integer, ByVal <i>Delay</i> As Double, ByVal <i>Algorithm</i> As Constant)
<b>Parameter</b>	See Appendix A for Settling Algorithm and parameter name descriptions.
<b>Description</b>	This command sets the settling parameters for the ATS.Aux.ReadingRdg command.
<b>See Also</b>	ATS.Aux.ReadingRdg, ATS.Aux.ReadingReady, ATS.Aux.ReadingTrig
<b>Example</b>	See example for ATS.Aux.ReadingRdg.

**ATS.Aux.ReadingTrig****Method**

<b>Syntax</b>	<b>ATS.Aux.ReadingTrig</b> (ByVal <i>Number</i> As Constant)	
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Number</i>	apbAux1 = Auxiliary Instrument 1 apbAux2 = Auxiliary Instrument 2 apbAux3 = Auxiliary Instrument 3 apbAux4 = Auxiliary Instrument 4
<b>Description</b>	Causes a restart of the reading cycle and zeros the ready count for the ATS.Aux.ReadingRdg command. The reading in progress is aborted.	
<b>See Also</b>	ATS.Aux.ReadingRdg, ATS.Aux.ReadingReady, ATS.Aux.ReadingSettling	
<b>Example</b>	See example for ATS.Aux.ReadingRdg.	

## ATS.Aux.SetReading

## Method

**Syntax** `ATS.Aux.SetReading (ByVal Number As Constant, ByVal Value as Variant)`

Parameter	Name	Description
	<i>Number</i>	apbAux1 = Auxiliary Instrument 1 apbAux2 = Auxiliary Instrument 2 apbAux3 = Auxiliary Instrument 3 apbAux4 = Auxiliary Instrument 4
	<i>Value</i>	For <i>Number</i> 1 and 2 the <i>Value</i> Data Type is Double. For <i>Number</i> 3 and 4 the <i>Value</i> Data Type is Long.

**Description** This command sets the value used by the Auxiliary Instrument Reading parameter.

**Example** See example for `ATS.Aux.ReadingRdg`.

## ATS.Aux.Setting

## Property

**Syntax** `ATS.Aux.Setting (ByVal Number As Constant)`

**Data Type** Double

Parameter	Name	Description
	<i>Number</i>	apbAux1 = Auxiliary Instrument 1 apbAux2 = Auxiliary Instrument 2 apbAux3 = Auxiliary Instrument 3 apbAux4 = Auxiliary Instrument 4

**Description** This command sets the value used by the Auxiliary Instrument Setting parameter.

**Example** See example for `ATS.Aux.ReadingRdg`.

User Notes

### ATS.BarGraph.AxisAutoScale

Property

**Syntax** `ATS.BarGraph.AxisAutoScale (ByVal BarId As Integer)`

**Data Type** Boolean

*True* Auto scale Bar Graph.  
*False* Disable auto scale.

**Parameter**

Name	Description
<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.

**Description** Set the selected Bar Graph Axis to Auto Scale.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AGen.Freq(apbChA, "Hz") = 3000.0
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.Inst.Analyzer.FuncMode = apbAnlrBandpass
    ATS2.Inst.Analyzer.FuncBPBRTuning = apbAnlrFixed
    ATS2.Inst.Analyzer.FuncBPBRFreq("Hz") = 3000.0
```

With **ATS.BarGraph**

```
    GenFreqBar = .New
    .Id(GenFreqBar) = 5051
    .AxisLogLin(GenFreqBar) = apbLin
    .AxisRight(GenFreqBar, "Hz") = 3500.0
    .AxisLeft(GenFreqBar, "Hz") = 2500.0
    .AxisIncrement(GenFreqBar, "Hz") = 10.0
    AnlrFuncRdg = .New
    .Id(AnlrFuncRdg) = 6345
```

```

.DigitsOnly(AnlrFuncRdg) = False
.AxisLogLin(AnlrFuncRdg) = apbLin
.AxisLeft(AnlrFuncRdg, "V") = 0.8
.AxisRight(AnlrFuncRdg, "V") = 1.2
.AxisAutoScale(AnlrFuncRdg) = True
.TargetLower(AnlrFuncRdg, "V") = 0.95
.TargetUpper(AnlrFuncRdg, "V") = 1.05
.TargetRange(AnlrFuncRdg) = True
.Reset(GenFreqBar)
.Reset(AnlrFuncRdg)

String1$ = "Adjust Generator Frequency using _
    Bargraph #" & GenFreqBar
String2$ = " for Maximum Amplitude on _
    Bargraph #" & AnlrFuncRdg & "."
ATS.Prompt.Text = String1$ & String2$
ATS.Prompt.Position(0,0,1150,120)
ATS.Prompt.ShowWithContinue
Stop

GenMaxSet = .Max(GenFreqBar)
GenMinSet = .Min(GenFreqBar)
AnlrMaxRdg = .Max(AnlrFuncRdg)
AnlrMinRdg = .Min(AnlrFuncRdg)
End With

MaxSet$ = "Maximum Frequency = " _
    & Left(Str$(GenMaxSet),6) & " Hz" & Chr(13)
MinSet$ = "Minimum Frequency = " _
    & Left(Str$(GenMinSet),6) & " Hz" & Chr(13) _
    & Chr$(13)
MaxRdg$ = "Maximum Voltage = " _
    & Left(Str$(AnlrMaxRdg),6) & " V" & Chr(13)
MinRdg$ = "Minimum Voltage = " _
    & Left(Str$(AnlrMinRdg),6) & " V" & Chr(13) _
    & Chr$(13)
CurSet$ = "Current Frequency Setting = " _
    & Left(Str$(Gen.Freq("Hz")),6) & " Hz" & Chr(13)
CurRdg$ = "Current Amplitude Reading = " _
    & Left(Str$(Anlr.FuncRdg("V")),6) & " V"

```

```

ATS.Prompt.Text = MaxSet$ & MinSet$ & MaxRdg$ _
                & MinRdg$ & CurSet$ & CurRdg$
ATS.Prompt.Position(0,0,550,350)
ATS.Prompt.ShowWithContinue
Stop
End Sub
    
```

## ATS.BarGraph.AxisIncrement

## Property

**Syntax** `ATS.BarGraph.AxisIncrement (ByVal BarId As Integer, ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.
	<i>Unit</i>	Refer to the setting or reading defined by the <code>ATS.BarGraph.Id</code> command to determine the appropriate unit selections.

**Description** Set the selected Bar Graph increment decrement size. When the Bar Graph is configured to control a setting (for example the generator frequency) the arrow keys can be used to increment or decrement the frequency by the increment value.

**Example** See example for `ATS.BarGraph.AxisAutoScale`.

## ATS.BarGraph.AxisLeft

## Property

**Syntax** `ATS.BarGraph.AxisLeft (ByVal BarId As Integer, ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.

*Unit* Refer to the setting or reading defined by the `ATS.BarGraph.Id` command to determine the appropriate unit selections.

**Description** This command defines the value on the left side of the Bar Graph.

**See Also** `ATS.BarGraph.AxisRight`,  
`ATS.BarGraph.AxisAutoScale`

**Example** See example for `ATS.BarGraph.AxisAutoScale`.

## ATS.BarGraph.AxisLogLin

## Property

**Syntax** `ATS.BarGraph.AxisLogLin (ByVal BarId As Integer)`

**Data Type** Constant

*apbLog*

Logarithmic axis.

*apbLin*

Linear axis.

**Parameter**

**Name**

**Description**

*BarId*

Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.

**Description** This command determines the Bar Graph axis data scaling type.

**Example** See example for `ATS.BarGraph.AxisAutoScale`.

## ATS.BarGraph.AxisRight

## Property

**Syntax** `ATS.BarGraph.AxisRight (ByVal BarId As Integer)`

**Data Type** Double

**Parameter**

**Name**

**Description**

*BarId*

Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.



*unit\$* Refer to the setting or reading defined by the `ATS.BarGraph.Id` command to determine the appropriate unit selections.

**Description** This command defines the value on the right side of the Bar Graph.

**See Also** `ATS.BarGraph.AxisLeft`,  
`ATS.BarGraph.AxisAutoScale`

**Example** See example for `ATS.BarGraph.AxisAutoScale`.

## ATS.BarGraph.Comment

## Property

**Syntax** `ATS.BarGraph.Comment (ByVal BarId As Integer)`

**Data Type** String ASCII characters.

Parameter	Name	Description
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.

**Description** This command transfers the ASCII characters to or from the comment section in the BarGraph panel to a string variable.

**See Also** `ATS.BarGraph.CommentShow`

**Example**

```

Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    With ATS2.Inst.Analyzer
        .FuncMode = apbAnlrBandpass
        .FuncBPBRTuning = apbAnlrFixed
        .FuncBPBRFreq("Hz") = 3000.0
    End With
    With ATS.BarGraph
        BarID1 = .New(6345)
        .AxisLeft(BarID1,"Hz") = 2500.0
        .AxisRight(BarID1,"Hz") = 3500.0
        .AxisIncrement(BarID1,"Hz") = 1.0
        .Title(BarID1) = "Bar Graph 1: _
            Analog Generator Frequency"
    End With
End Sub
    
```

```

BarID2 = .New(5907)
.AxisLeft(BarID2,"V") = 0.50
.AxisRight(BarID2,"V") = 1.50
.CommentShow(BarID2) = True
.Title(BarID2) = "Analog Analyzer Bandpass _
    Amplitude"
.Comment(BarID2) = "Adjust Bar Graph #1 for _
    maximum amplitude reading."
End With
With ATS.Prompt
    .FontSize = 8
    .Position(290,244,225,120)
    .Text = Chr$(10) & "Press this button to _
        proceed."
    .ShowWithContinue
Stop
End With
Debug.Print "Filter peek = " & _
    ATS2.AGen.Freq("Hz") & " Hz"
End Sub

```

---

## ATS.BarGraph.CommentShow

## Property

<b>Syntax</b>	ATS.BarGraph.CommentShow
<b>Data Type</b>	Boolean
	<i>True</i> Display Comment section.
	<i>False</i> Remove Comment section from view.
<b>Description</b>	This command displays or removes from view the comment section in the Graph panel
<b>See Also</b>	ATS.BarGraph.Comment
<b>Example</b>	See example for ATS.BarGraph.Comment.

## ATS.BarGraph.DigitsOnly

Property

**Syntax** `ATS.BarGraph.DigitsOnly (ByVal BarId As Integer)`

**Data Type** Boolean

<i>True</i>	Display digits only.
<i>False</i>	Display digits and Bar Graph.

Parameter	Name	Description
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.

**Description** This command displays only the digits (numeric characters) or the digits and the bar on the Bargraph.

**Example** See example for `ATS.BarGraph.AxisAutoScale`.

## ATS.BarGraph.Id

Property

**Syntax** `ATS.BarGraph.Id (ByVal BarId As Integer)`

**Data Type** Long Instrument Parameter ID#.

Parameter	Name	Description
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.

**Description** This command is used to select the instrument parameter, which will return readings or control settings for the selected Bar Graph.

Refer to Appendix B to obtain instrument parameter identification numbers.

**Example** See example for `ATS.BarGraph.AxisAutoScale`.

## ATS.BarGraph.Max

Property

**Syntax** `ATS.BarGraph.Max (ByVal BarId As Integer, ByVal Unit As String)`

Parameter	Name	Description
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.
	<i>Unit</i>	Refer to the setting or reading defined by the <code>ATS.BarGraph.Id</code> command to determine the appropriate unit selections.
<b>Result</b>	Double	
<b>Description</b>	This command returns the maximum measured value obtained during the time since the last reset for the selected Bar Graph	
<b>See Also</b>	<code>ATS.BarGraph.Reset</code> , <code>ATS.BarGraph.Min</code>	
<b>Example</b>	See example for <code>ATS.BarGraph.AxisAutoScale</code> .	

## ATS.BarGraph.Min

### Property

**Syntax** `ATS.BarGraph.Min` (ByVal *BarId* As Integer, ByVal *Unit* As String)

Parameter	Name	Description
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.
	<i>Unit</i>	Refer to the setting or reading defined by the <code>ATS.BarGraph.Id</code> command to determine the appropriate unit selections.

**Result** Double

**Description** This command returns the minimum measured value obtained during the time since the last reset for the selected Bar Graph

**See Also** `ATS.BarGraph.Reset`, `ATS.BarGraph.Max`

**Example** See example for `ATS.BarGraph.AxisAutoScale`.

## ATS.BarGraph.New

### Method

**Syntax** `ATS.BarGraph.New` [ (Optional ByVal *ConId* As Variant) ]

Parameter	Name	Description
	<i>ConId</i>	Instrument identification number. Refer to Appendix B to obtain instrument parameter identification numbers.
<b>Result</b>	Integer	
	<i>1-32</i>	Identification number of Bar Graph created.
<b>Description</b>	This command creates a new Bar Graph and returns the identification number.	
<b>Example</b>	See example for <code>ATS.BarGraph.AxisAutoScale</code> .	

## ATS.BarGraph.Reset

**Method**

**Syntax** `ATS.BarGraph.Reset (ByVal BarId As Integer)`

Parameter	Name	Description
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.

**Description** This command resets the selected Bar Graph. The reset action sets the Min and Max. values to the current reading and as additional readings are taken the Min and Max. readings track the deviations

**See Also** `ATS.BarGraph.Max`, `ATS.BarGraph.Min`

**Example** See example for `ATS.BarGraph.AxisAutoScale`.

## ATS.BarGraph.TargetLower

**Property**

**Syntax** `ATS.BarGraph.TargetLower (ByVal BarId As Integer, ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.

	<i>Unit</i>	Refer to the setting or reading defined by the <code>ATS.BarGraph.Id</code> command to determine the appropriate unit selections.
<b>Description</b>	This command defines the target value for the left side of the Bar Graph.	
<b>See Also</b>	<code>ATS.BarGraph.TargetUpper</code> , <code>ATS.BarGraph.TargetRange</code>	
<b>Example</b>	See example for <code>ATS.BarGraph.AxisAutoScale</code> .	

---

## ATS.BarGraph.TargetRange

## Property

<b>Syntax</b>	<b>ATS.BarGraph.TargetRange</b> (ByVal <i>BarId</i> As Integer)	
<b>Data Type</b>	Boolean	
	<i>True</i>	Target area displayed.
	<i>False</i>	Target area not displayed.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.
<b>Description</b>	This command turns the selected Bar Graph Target Range ON or OFF.	
<b>See Also</b>	<code>ATS.BarGraph.TargetLower</code> , <code>ATS.BarGraph.TargetUpper</code>	
<b>Example</b>	See example for <code>ATS.BarGraph.AxisAutoScale</code> .	

---

## ATS.BarGraph.TargetUpper

## Property

<b>Syntax</b>	<b>ATS.BarGraph.TargetUpper</b> (ByVal <i>BarId</i> As Integer, ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	

Parameter	Name	Description
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.
	<i>Unit</i>	Refer to the setting or reading defined by the <code>ATS.BarGraph.Id</code> command to determine the appropriate unit selections.
<b>Description</b>	This command defines the value on the right side of the Bar Graph.	
<b>See Also</b>	<code>ATS.BarGraph.TargetLower</code> , <code>ATS.BarGraph.TargetRange</code>	
<b>Example</b>	See example for <code>ATS.BarGraph.AxisAutoScale</code> .	

## ATS.BarGraph.Title

## Property

<b>Syntax</b>	<code>ATS.BarGraph.Title</code> (ByVal <i>BarId</i> As Integer)	
<b>Data Type</b>	String	ASCII characters.
Parameter	Name	Description
	<i>BarId</i>	Bar Graph identification number (1-32). The identification number is located on the Bar Graph title bar.
<b>Description</b>	This command transfers the ASCII characters to or from the title bar in the BarGraph panel to a string variable.	
<b>Example</b>	See example for <code>ATS.BarGraph.Comment</code> .	

User Notes



---

### ATS.Comm.Break

Property

**Syntax**      `ATS.Comm (ByVal Num as Integer) .Break`

**Data Type**      Boolean

*True*              Sets the break signal.  
*False*             Clears the break signal.

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description**      This command sets or clears the break signal. Setting the break signal to True stops sending characters and places the line in a break state until the Break command is set to False.

---

### ATS.Comm.CD Holding

Property

**Syntax**      `ATS.Comm (ByVal Num as Integer) .CDHolding`

**Data Type**      Boolean

*True*              Carrier Detect line high.  
*False*             Carrier Detect line low.

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description**      This command returns the state of the Carrier Detect (CD) line. The state of the Carrier Detect line indicates to the computer whether or not the modem is online.

When the Carrier Detect line is high (CDHolding = True) and the time specified by the `ATS.Comm.CDTimeout` command has expired, the `ATS.Comm.CommError` command is set to `comCDTO` (Carrier Detect Timeout Error).

The Carrier Detect is also known as the Receive Line Signal Detect (RLSD).

**See Also** `ATS.Comm.CDTimeout`

---

## ATS.Comm.CDTimeout

**Property**

**Syntax** `ATS.Comm (ByVal Num As Integer) .CDTimeout`

**Data Type** Long

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command sets and returns the maximum amount of time (in milliseconds) that the control waits for the Carrier Detect (CD) signal before timing out. This command indicates a timeout condition by setting the `ATS.Comm.CommError` command to `CDTO` (Carrier Detect Timeout Error).

---

## ATS.Comm.CommError

**Property**

**Syntax** `Event = ATS.Comm (ByVal Num As Integer) .CommError`

**Data Type** Integer

The following list contains communications errors or events.

Setting	Value	Description
<i>comBreak</i>	1001	Break signal received.
<i>comCTSTO</i>	1002	Clear To Send Timeout. The Clear To Send line was low for the number of milliseconds specified

		by the ATS.Comm.CTSTimeout command while trying to send a character.
<i>comDSRTO</i>	1003	Data Set Ready Timeout. The Data Set Ready line was low for the number of milliseconds specified by the ATS.Comm.DSRTimeout command while trying to send a character.
<i>comFrame</i>	1004	Framing Error. The hardware detected a framing error.
<i>comOverrun</i>	1006	Port Overrun. A character was not read from the hardware before the next character arrived and was lost.
<i>comCDTO</i>	1007	Carrier Detect Timeout. The Carrier Detect line was low for the number of milliseconds specified by the ATS.Comm.CDTimeout command while trying to send a character.
<i>comRxOver</i>	1008	Receive Buffer Overflow. The receive buffer is full.
<i>comRxParity</i>	1009	Parity Error. Parity error detected.
<i>comTxFull</i>	1010	Transmit Buffer Full. The transmit buffer was full while trying to queue a character.

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.
<b>Description</b>	Returns the most recent communication error.	

---

**ATS.Comm.CommId** **Property**

**Syntax**      **ATS.Comm** (ByVal *Num* As Integer) .**CommId**

**Data Type**    Integer

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description**    This command returns a handle that identifies the communications device.

**ATS.Comm.CommPort****Property**

**Syntax** `ATS.Comm (ByVal Num As Integer) .CommPort`

**Data Type** Integer

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command sets and returns the communications port number. The communications control generates error 68 (Device unavailable) if the port does not exist.

Warning You must set `ATS.Comm.CommPort` before opening the port.

**Example**

```
Sub Main
    If ATS.Comm(1).PortOpen = True Then
        ATS.Comm(1).PortOpen = False
    End If

    'Port Setup
    ATS.Comm(1).CommPort = 2
    ATS.Comm(1).Settings = "9600,N,8,1"
    ATS.Comm(1).OutBufferSize = 10
    ATS.Comm(1).InBufferSize = 10

    'Output to Comm Port 2
    ATS.Comm(1).PortOpen = True
    ATS.Comm(1).Output = "1234567890"

    'Input from Comm Port 2
    Character$ = ATS.Comm(1).Input
    Debug.Print Character$

    ATS.Comm(1).PortOpen = False
End Sub
```

## ATS.Comm.CTSHolding

## Property

**Syntax** `ATS.Comm (ByVal Num As Integer) .CTSHolding`

**Data Type** Boolean

*True* Clear To Send line high.  
*False* Clear To Send line low.

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command returns the state of the of the Clear To Send (CTS) line. The state of the Clear To Send line indicates to the computer whether or not the transmission can proceed.

When the Clear To Send line is low (CTSHolding = False) and the time specified by the `ATS.Comm.CTSTimeout` command has expired, the `ATS.Comm.CommError` command is set to `comCTSTO` (Clear To Send Timeout).

The Clear To Send line is used in RTS/CTS (Request To Send/Clear To Send) hardware handshaking. The `ATS.Comm.CTSHolding` command provides a way to manually determine the state of the Clear To Send line.

**See Also** `ATS.Comm.Handshaking`

## ATS.Comm.CTSTimeout

## Property

**Syntax** `ATS.Comm (ByVal Num As Integer) .CTSTimeout`

**Data Type** Long

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command sets and returns the maximum amount of time (in milliseconds) that the control waits for the Clear To Send (CTS) signal before timing out. This command indicates a timeout condition

by setting the `ATS.Comm.CommError` command to CTSTO (Clear To Send Timeout Error).

---

## ATS.Comm.DSRHolding

### Property

**Syntax** `ATS.Comm (ByVal Num As Integer) .DSRHolding`

**Data Type** Boolean

*True* Data Set Ready line high.  
*False* Data Set Ready line low.

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command returns the state of the of the Data Set Ready (DSR) line. The state of the Data Set Ready line indicates to the computer whether or not the hardware is ready to proceed.

---

## ATS.Comm.DSRTIMEOUT

### Property

**Syntax** `ATS.Comm (ByVal Num As Integer) .DSRTIMEOUT`

**Data Type** Long

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command sets and returns the maximum amount of time (in milliseconds) that the control waits for the Data Set Ready (DSR) signal before timing out. This command indicates a timeout condition by setting the `ATS.Comm.CommError` command to DSRTO (Data Set Ready Timeout Error).

---

## ATS.Comm.DTRENABLE

### Property

**Syntax** `ATS.Comm (ByVal Num As Integer) .DTRENABLE`

<b>Data Type</b>	Boolean	
	<i>True</i>	Enable the Data Terminal Ready (line high) when port opened and (line Low) when the port is closed.
	<i>False</i>	(Default) Disable the Data Terminal Ready (line always low).
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.
<b>Description</b>	<p>This command determines whether to enable the Data Terminal Ready (DTR) line during communications. Typically, the Data Terminal Ready signal is sent by a computer to its modem to indicate that the computer is ready to accept incoming data.</p> <p>Setting the Data Terminal Ready line to low in most cases hangs up the telephone.</p>	

## ATS.Comm.Handshaking

## Property

<b>Syntax</b>	<b>ATS . Comm (ByVal <i>Num</i> As Integer) . Handshaking</b>	
<b>Data Type</b>	Constant	
	<i>apbComNone</i>	(Default) No handshaking.
	<i>apbComXON_XOFF</i>	XON/XOFF handshaking.
	<i>apbComRTS_CTS</i>	RTS/CTS (Request To Send/Clear To Send) handshaking.
	<i>apbComSendXON_XOFF</i>	Both Request To Send and XON/XOFF handshaking.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.
<b>Description</b>	This command sets and returns the state of the hardware handshaking.	

Handshaking refers to the internal communications protocol by which data is transferred from the hardware port to the receive buffer. When a character of data arrives at the serial port, the communications device has to move it into the receive buffer so that your program can read it. If there is no receive buffer and your program is expected to read every character directly from the hardware, you will probably lose data because the characters can arrive very quickly.

A handshaking protocol insures that data is not lost due to a buffer overrun, in which case data arrives at the port too quickly for the communications device to move the data into the receive buffer.

---

## ATS.Comm.InBufferCount

## Property

**Syntax** `ATS.Comm (ByVal Num As Integer) .InBufferCount`

**Data Type** Integer

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command returns the number of characters in the receive buffer.

---

## ATS.Comm.InBufferSize

## Property

**Syntax** `ATS.Comm (ByVal Num As Integer) .InBufferSize`

**Data Type** Integer

**Description** This command sets and returns the size of the receive buffer in bytes. The default receive buffer size is 1024.

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Example** See example for `ATS.Comm.CommPort`.



## ATS.Comm.Input

**Property****Syntax** `ATS.Comm (ByVal Num As Integer) .Input`**Result** String

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command returns and removes a string of characters from the receive buffer. The `ATS.Comm.InputLen` command defines the number of characters that are read by the `ATS.Comm.Input` command.

**Example** See example for `ATS.Comm.CommPort`.

## ATS.Comm.InputLen

**Property****Syntax** `ATS.Comm (ByVal Num As Integer) .InputLen`**Data Type** Integer

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command sets and returns the number of characters the `ATS.Comm.Input` command reads from the receive buffer.

Setting the `ATS.Comm.InputLen` command to 0 causes the `ATS.Comm.Input` command to read the entire contents of the receive buffer.

If `InputLen` characters are not available in the receive buffer, the `ATS.Comm.Input` command returns a zero-length string (""). The `ATS.Comm.InBufferCount` command can also be checked to determine if the required number of characters are present before using the `ATS.Comm.Input` command.

**ATS.Comm.Interval****Property**

**Syntax**      `ATS.Comm (ByVal Num As Integer) .Interval`

**Data Type**      Long

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description**      This command sets the interval (milliseconds) for polling the hardware for data under the Windows 3.0 operating system.

**ATS.Comm.NullDiscard****Property**

**Syntax**      `ATS.Comm (ByVal Num As Integer) .NullDiscard`

**Data Type**      Boolean

<i>True</i>	Null characters are not transferred from the port to the receive buffer.
<i>False</i>	(Default) Null characters are transferred from the port to the receive buffer.

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description**      This command determines whether null characters are allowed into the receive buffer.

A null character is defined as ASCII character 0, Chr\$(0).

**ATS.Comm.OutBufferCount****Property**

**Syntax**      `ATS.Comm (ByVal Num As Integer) .OutBufferCount`

**Data Type**      Integer

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.
<b>Description</b>	This command returns the number of characters in the transmit buffer. The transmit buffer can be cleared by setting the <code>ATS.Comm.OutBufferSize</code> command to 0.	

---

## ATS.Comm.OutBufferSize Property

<b>Syntax</b>	<code>ATS.Comm (ByVal Num As Integer) .OutBufferSize</code>	
<b>Data Type</b>	Integer	
Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.
<b>Description</b>	This command sets and returns the size, in characters, of the transmit buffer. The default transmit buffer size is 512 bytes.	
<b>Example</b>	See example for <code>ATS.Comm.CommPort</code> .	

---

## ATS.Comm.Output Property

<b>Syntax</b>	<code>ATS.Comm (ByVal Num As Integer) .Output</code>	
<b>Data Type</b>	Variant	
Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.
<b>Description</b>	This command sends a string of characters to the transmit buffer.	
<b>Example</b>	See example for <code>ATS.Comm.CommPort</code> .	

**ATS.Comm.ParityReplace****Property**

**Syntax** `ATS.Comm (ByVal Num As Integer) .ParityReplace`

**Data Type** String

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command sets and returns the character that replaces an invalid character in the data if a parity error occurs.

The parity bit refers to a bit that is transmitted along with a specified number of data bits to provide error checking. When you use a parity bit, the communications control adds up all the bits that are set (having a value of 1) in the data and tests the sum as being odd or even (according to the parity setting used when the port was opened).

By default, the control uses a question mark (?) character for replacing invalid characters. Setting ParityReplace to an empty string ("") disables replacement of the character where the parity error occurs.

**ATS.Comm.PortOpen****Property**

**Syntax** `ATS.Comm (ByVal Num As Integer) .PortOpen`

**Data Type** Boolean

<i>True</i>	Port is opened.
<i>False</i>	Port is closed or closes the port and clears the receive and transmit buffers.

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command sets and returns the state of the communications port.

If either the `ATS.Comm.DTREnable` or the `ATS.Comm.RTSEnable` commands are set to `True` before the port is opened, the state of each command is set to `False` when the port is closed. Otherwise, the DTR and RTS lines remain in their previous state.

**Example** See example for `ATS.Comm.CommPort`.

---

## ATS.Comm.RTSEnable

## Property

**Syntax** `ATS.Comm (ByVal Num As Integer) .RTSEnable`

**Data Type** Boolean

<i>True</i>	Enables the Request To Send line (line set high when port open and low when port closed).
<i>False</i>	The default condition, disables the Request To Send line.

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command determines the state of the Request To Send line. The Request To Send line is used in RTS/CTS hardware handshaking.

---

## ATS.Comm.Settings

## Property

**Syntax** `ATS.Comm (ByVal Num As Integer) .Settings`

**Data Type** String

The following table lists the valid baud rates.

Setting	Description
<i>110</i>	
<i>300</i>	
<i>600</i>	
<i>1200</i>	

2400  
 9600 (Default)  
 14400  
 19200  
 38400  
 57600  
 11500  
 230400  
 460800  
 921600

The following table describes the valid parity values.

Setting	Description
<i>E</i>	Even
<i>M</i>	Mark
<i>N</i>	None (Default)
<i>O</i>	Odd
<i>S</i>	Space

The following table lists the valid data bit values.

Setting	Description
4	
5	
6	
7	
8	(default)

The following table lists the valid stop bit values.

Setting	Description
1	(Default)
1.5	
2	

Parameter	Name	Description
	<i>Num</i>	1 or 2 Two RS-232 communication channels can be defined at one time.

**Description** This command sets and returns the baud rate, parity, data bit, and stop bit settings.

If paramString\$ is not valid when the port is opened, the communications control generates error 380 (Invalid property value).

Settings\$ consists of four parts as specified in the following format:

"B,P,D,S"

Part	Description
<i>B</i>	Baud rate
<i>P</i>	Parity
<i>D</i>	Number of data bits
<i>S</i>	Number of stop bits

The default value of Settings\$ is: "9600,N,8,1"

**Example** See example for `ATS.Comm.CommPort.`

User Notes



### ATS.Compute.Avg.Apply

Method

**Syntax**            `ATS . Compute . Avg . Apply`

**Result**            Boolean

*True*                Computation performed.  
*False*              Computation NOT performed.

**Description**    This command applies the Average computation to the selected data (1-6). All of the measurements in the selected data will be replaced with the average value of the data within the Start and Stop settings for the Compute Average function.

**See Also**        `ATS . Compute . Clear . All`, `ATS . Compute . Avg . Data`,  
`ATS . Compute . Avg . PostSweep`,  
`ATS . Compute . Avg . Start`, `ATS . Compute . Avg . Stop`,

**Example**        `Sub Main`  
                  `ATS . Application . NewTest`  
                  `ATS2 . AGen . OutputOn = True`  
                  `ATS2 . AnalogIn . Source (apbChA) = apbAnalogInGenMon`  
                  `ATS . Compute . Avg . Data (1) = True`  
                  `ATS . Compute . Avg . PostSweep = False`  
                  `ATS . Compute . Avg . Start ("Hz") = 5000`  
                  `ATS . Compute . Avg . Stop ("Hz") = 100`  
                  `ATS . Sweep . Start`  
                  `ATS . Compute . Avg . Apply`  
                  `End Sub`

**ATS.Compute.Avg.Data****Property**

**Syntax** `ATS.Compute.Avg.Data (ByVal Source As Integer)`

**Data Type** Boolean

*True* Select specified data.  
*False* Deselect specified data.

**Parameter**

Name	Description
<i>Source</i>	1 = Data 1 measurements 2 = Data 2 measurements 3 = Data 3 measurements 4 = Data 4 measurements 5 = Data 5 measurements 6 = Data 6 measurements

**Description** This command determines which data (1-6) the Average computation is to be performed on. By using this command several times to select multiple data sources, several Average computations can be performed in one operation.

**See Also** `ATS.Compute.Avg.Apply`

**Example** See example for `ATS.Compute.Avg.Apply`.

**ATS.Compute.Avg.PostSweep****Property**

**Syntax** `ATS.Compute.Avg.PostSweep`

**Data Type** Boolean

*True* Enable computation to be applied after sweep.  
*False* Disable computation after sweep.

**Description** This command instructs the test to perform the Average computation after a sweep is complete and sets the state of the Apply After Sweep field on the Compute Average panel.

ATS retains the order in which the Apply After Sweep field on the Compute panels is enabled. This permits multiple computations to be

performed on data from a single test. The order of the computations is also retained in the test file.

**See Also** `ATS.Compute.Avg.Apply`

**Example** See example for `ATS.Compute.Avg.Apply`.

## ATS.Compute.Avg.Start

**Property**

**Syntax** `ATS.Compute.Avg.Start (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.

**Description** This command sets the Start value of the data over which the Average computation will be performed.

**See Also** `ATS.Compute.Avg.Stop`, `ATS.Compute.Avg.Apply`

**Example** See example for `ATS.Compute.Avg.Apply`.

## ATS.Compute.Avg.StartUnit

**Method**

**Syntax** `ATS.Compute.Avg.StartUnit`

**Result** String

**Description** This command returns the Unit used for the Start setting for the Compute Average function.

**See Also** `ATS.Compute.Avg.StopUnit`

**Example**

```
Sub Main
    ATS.Compute.Avg.Data(1) = True
    ATS.Compute.Avg.Start("Hz") = 20000.0
    Debug.Print ATS.Compute.Avg.StartUnit
End Sub
```

**Output** Hz

---

## ATS.Compute.Avg.Stop

**Property**

**Syntax** `ATS.Compute.Avg.Stop (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.

**Description** This command sets the Stop value of the data over which the Average computation will be performed.

**See Also** `ATS.Compute.Avg.Start`, `ATS.Compute.Avg.Apply`

**Example** See example for `ATS.Compute.Avg.Apply`.

---

## ATS.Compute.Avg.StopUnit

**Method**

**Syntax** `ATS.Compute.Avg.StopUnit`

**Result** String

**Description** This command returns the Unit used for the Stop setting for the Compute Average function.

**See Also** `ATS.Compute.Avg.StartUnit`

**Example**

```
Sub Main
    ATS.Compute.Avg.Data(1) = True
    ATS.Compute.Avg.Stop("Hz") = 20000.0
    Debug.Print ATS.Compute.Avg.StopUnit
End Sub
```

**Output** Hz

## ATS.Compute.Center.Apply

Method

**Syntax**            **ATS . Compute . Center . Apply**

**Result**            Boolean

*True*                Computation performed.  
*False*              Computation NOT performed.

**Description**    This command applies the Center computation to the selected data (1-6).

**See Also**        *ATS.Compute.Clear.All, ATS.Compute.Center.Data, ATS.Compute.Center.PostSweep, ATS.Compute.Center.Start, ATS.Compute.Center.Stop*

**Example**

```
Sub Main
  ATS.File.OpenTest "CENTER.ATS2"
  ATS . Compute . Clear . All
  ATS . Compute . Center . Data(1) = True
  ATS . Compute . Center . PostSweep = False
  ATS . Compute . Center . Start("Hz") = 200000
  ATS . Compute . Center . Stop("Hz") = 10
  ATS.Sweep.Start
  ATS . Compute . Center . Apply
End Sub
```

## ATS.Compute.Center.Data

Property

**Syntax**            **ATS . Compute . Center . Data** (ByVal *Source* As Integer)

**Data Type**        Boolean

*True*                Select specified data.  
*False*              Deselect specified data.

**Parameter**

Name	Description
<i>Source</i>	1 = Data 1 measurements 2 = Data 2 measurements 3 = Data 3 measurements 4 = Data 4 measurements

5 = Data 5 measurements

6 = Data 6 measurements

**Description** This command determines which data (1-6) the Center computation is to be performed on. By using this command several times to select multiple data sources, several Center computations can be performed in one operation.

**See Also** `ATS.Compute.Center.Apply`

**Example** See example for `ATS.Compute.Center.Apply`.

## ATS.Compute.Center.PostSweep

### Property

**Syntax** `ATS.Compute.Center.PostSweep`

**Data Type** Boolean

*True* Enable computation to be applied after sweep.  
*False* Disable computation after sweep.

**Description** This command instructs the test to perform the Center computation after a sweep is complete and sets the state of the Apply After Sweep field on the ComputeCenter panel.

ATS retains the order in which the Apply After Sweep field on the Compute panels is enabled. This permits multiple computations to be performed on data from a single test. The order of the computations is also retained in the test file.

**See Also** `ATS.Compute.Center.Apply`

**Example** See example for `ATS.Compute.Center.Apply`.

## ATS.Compute.Center.Start

### Property

**Syntax** `ATS.Compute.Center.Start (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.
<b>Description</b>	This command sets the Start value of the data over which the Center computation will be performed.	
<b>See Also</b>	ATS.Compute.Center.Stop, ATS.Compute.Center.Apply	
<b>Example</b>	See example for ATS.Compute.Center.Apply.	

---

## ATS.Compute.Center.StartUnit Method

<b>Syntax</b>	<code>ATS.Compute.Center.StartUnit</code>
<b>Result</b>	String
<b>Description</b>	This command returns the Unit used for the Start setting for the Compute Center function.
<b>See Also</b>	ATS.Compute.StopUnit
<b>Example</b>	<pre>Sub Main     ATS.Compute.Center.Data(1) = True     ATS.Compute.Center.Start("Hz") = 0.02     Debug.Print <b>ATS.Compute.Center.StartUnit</b> End Sub</pre>
<b>Output</b>	Hz

---

## ATS.Compute.Center.Stop Property

<b>Syntax</b>	<code>ATS.Compute.Center.Stop (ByVal Unit As String)</code>
<b>Data Type</b>	Double

Parameter	Name	Description
	<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.
<b>Description</b>		This command sets the Stop value of the data over which the Center computation will be performed.
<b>See Also</b>		ATS.Compute.Center.Start, ATS.Compute.Center.Apply
<b>Example</b>		See example for ATS.Compute.Center.Apply.

---

## ATS.Compute.Center.StopUnit

**Method**

<b>Syntax</b>	<b>ATS.Compute.Center.StopUnit</b>
<b>Result</b>	String
<b>Description</b>	This command returns the Unit used for the Stop setting for the Compute Average function.
<b>See Also</b>	ATS.Compute.Center.StartUnit
<b>Example</b>	<pre>Sub Main   ATS.Compute.Center.Data(1) = True   ATS.Compute.Center.Stop("F/R") = 20000.0   Debug.Print <b>ATS.Compute.Center.StopUnit</b> End Sub</pre>
<b>Output</b>	F/R

---

## ATS.Compute.Clear.All

**Method**

<b>Syntax</b>	<b>ATS.Compute.Clear.All</b>
<b>Description</b>	This command clears all computes from the current test.
<b>Example</b>	See example for ATS.Compute.Center.Apply.



## ATS.Compute.Delta.Apply

Method

<b>Syntax</b>	<b>ATS.Compute.Delta.Apply</b>
<b>Result</b>	<p>Boolean</p> <p><i>True</i>                    Computation performed.</p> <p><i>False</i>                    Computation NOT performed.</p>
<b>Description</b>	This command applies the Delta computation to the selected data (1-6).
<b>See Also</b>	ATS.Compute.Clear.All, ATS.Compute.Delta.Data, ATS.Compute.Delta.FileName, ATS.Compute.Delta.PostSweep
<b>Example</b>	<pre> Sub Main   ATS.File.OpenTest "DELTA.ATS2"   ATS.Sweep.Start   <b>ATS.Compute.Delta.FileName</b> = "DELTA1.ATSA"   <b>ATS.Compute.Delta.PostSweep</b> = False   <b>ATS.Compute.Delta.Data</b>(1,1) = True   <b>ATS.Compute.Delta.Apply</b> End Sub         </pre>

## ATS.Compute.Delta.Data

Property

<b>Syntax</b>	<b>ATS.Compute.Delta.Data</b> (ByVal <i>Source</i> As Integer, ByVal <i>Column</i> As Integer)	
<b>Data Type</b>	Boolean	
	<i>True</i>	Select specified data.
	<i>False</i>	Deselect specified data.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Source</i>	Number of the Sweep Data (1-6) of the data in memory.
	<i>Column</i>	Number of the Data Column (0-7) of the data specified by the <code>ATS.Compute.Delta.FileName</code> command.
<b>Description</b>	This command determines which data (Data 1-6) in memory and which data (Column 0-7) as specified by the	

`ATS.Compute.Delta.FileName` command the Delta computation is to be performed on. By using this command several times to select multiple data sources, several Delta computations can be performed in one operation.

**See Also** `ATS.Compute.Delta.Apply`,  
`ATS.Compute.Delta.FileName`

**Example** See example for `ATS.Compute.Delta.Apply`.

---

## ATS.Compute.Delta.DataColumn

## Method

**Syntax** `ATS.Compute.Delta.DataColumn (ByVal Data As Integer)`

**Data Type** Integer

Parameter	Name	Description
	<i>Data</i>	Number of the Sweep Data (1-6) of the data in memory.

**Description** This command returns the column in the attached file used in the Delta computation for the specified Sweep Data.

**See Also** `ATS.Compute.Delta.Data`,  
`ATS.Compute.Delta.FileName`

**Example**

```
Sub Main
    ATS.File.OpenTest "Delta Data Column.ats2"
    Debug.Print "Compute Delta using column #" & _
        & ATS.Compute.Delta.DataColumn(1)
End Sub
```

**Output** Compute Delta using column #1

---

## ATS.Compute.Delta.FileName

## Property

**Syntax** `ATS.Compute.Delta.FileName`

**Data Type** String Any valid DOS filename and extension. Enter "SweepData" for the file name to select data in memory.

**Description** This command attaches a data file to be used in the Compute Delta computation. The difference between the selected column data values in the data file and the selected data in memory will be calculated and then replace the data in memory.

**See Also** `ATS.Compute.Delta.Apply`

**Example** See example for `ATS.Compute.Delta.Apply`.

## ATS.Compute.Delta.PostSweep

Property

**Syntax** `ATS.Compute.Delta.PostSweep`

**Data Type** Boolean

*True* Enable computation to be applied after sweep.  
*False* Disable computation after sweep.

**Description** This command instructs the test to perform the Delta computation after a sweep is complete and sets the state of the Apply After Sweep field on the Compute Delta panel.

ATS retains the order in which the Apply After Sweep field on the Compute panels is enabled. This permits multiple computations to be performed on data from a single test. The order of the computations is also retained in the test file.

**See Also** `ATS.Compute.Delta.Apply`

**Example** See example for `ATS.Compute.Delta.Apply`.

## ATS.Compute.Equalize.Apply

Method

**Syntax** `ATS.Compute.Equalize.Apply`

**Result** Boolean

*True* Computation performed.  
*False* Computation NOT performed.

**Description** This command applies equalization to the selected data (1-6).

**See Also** `ATS.Compute.Clear.All,`  
`ATS.Compute.Equalize.Data,`  
`ATS.Compute.Equalize.FileName,`  
`ATS.Compute.Equalize.PostSweep`

**Example**

```
Sub Main
    ATS.File.OpenTest "EQ.ATS2"
    ATS.Sweep.Start
    ATS.Compute.Equalize.FileName = "EQ1.ATSA"
    ATS.Compute.Equalize.PostSweep = False
    ATS.Compute.Equalize.Data(1,1) = True
    ATS.Compute.Equalize.Data(2,2) = True
    ATS.Compute.Equalize.Apply
End Sub
```

---

## ATS.Compute.Equalize.Data

## Property

**Syntax** `ATS.Compute.Equalize.Data` (ByVal **Source** As Integer, ByVal **Column** As Integer)

**Data Type** Boolean

*True* Select specified data.  
*False* Deselect specified data.

Parameter	Name	Description
	<i>Source</i>	Number of the Sweep Data (1-6) of the data in memory.
	<i>Column</i>	Number of the Data Column (0-7) of the data specified by the <code>ATS.Compute.Equalize.FileName</code> command.

**Description** This command determines which data (Data 1-6) in memory and which data (Column 0-7) as specified by the `ATS.Compute.Equalize.FileName` command the Equalization computation is to be performed on. By using this command several times to select multiple data sources, several Equalization computations can be performed in one operation.

**See Also** `ATS.Compute.Equalize.Apply,`  
`ATS.Compute.Equalize.FileName`

**Example** See example for `ATS.Compute.Equalize.Apply`.

---

## ATS.Compute.Equalize.DataColumn Method

**Syntax** `ATS.Compute.Equalize.DataColumn (ByVal Data As Integer)`

**Data Type** Integer

Parameter	Name	Description
	<i>Data</i>	Number of the Sweep Data (1-6) of the data in memory.

**Description** This command returns the column in the attached file used in the Equalize computation for the specified Sweep Data.

**See Also** `ATS.Compute.Equalize.Data`,  
`ATS.Compute.Equalize.FileName`

**Example**

```
Sub Main
    ATS.File.OpenTest "Equalize Data Column.ats2"

    Debug.Print "Compute Equalize using EQ File " _
        & ATS.Compute.Equalize.FileName
    Debug.Print "Compute Equalize using column #" _
        & ATS.Compute.Equalize.DataColumn(1)
End Sub
```

**Output**

```
Compute Equalize using EQ File 50US-de.atsq
Compute Equalize using column #1
```

---

## ATS.Compute.Equalize.FileName Property

**Syntax** `ATS.Compute.Equalize.FileName`

**Data Type** String Any valid DOS filename and extension. Enter "SweepData" for the file name to select data in memory.

**Description** This command attaches a data file (Eq) to be used in the Compute Equalize computation. The data in memory is multiplied by the data in the Eq file.

**See Also** `ATS.Compute.Equalize.Apply`

**Example** See example for `ATS.Compute.Equalize.Apply`.

---

## ATS.Compute.Equalize.PostSweep

**Property**

**Syntax** `ATS.Compute.Equalize.PostSweep`

**Data Type** Boolean

*True* Enable computation to be applied after sweep.  
*False* Disable computation after sweep.

**Description** This command instructs the test to perform equalization after a sweep is complete and sets the state of the Apply After Sweep field on the Compute Equalize panel.

ATS retains the order in which the Apply After Sweep field on the Compute panels is enabled. This permits multiple computations to be performed on data from a single test. The order of the computations is also retained in the test file.

**See Also** `ATS.Compute.Equalize.Apply`

**Example** See example for `ATS.Compute.Equalize.Apply`.

---

## ATS.Compute.Invert.Apply

**Method**

**Syntax** `ATS.Compute.Invert.Apply`

**Result** Boolean

*True* Computation performed.  
*False* Computation NOT performed.

**Description** This command applies the Invert computation to the selected data (1-6).

**See Also**      `ATS.Compute.Clear.All`, `ATS.Compute.Invert.Data`,  
`ATS.Compute.Invert.Horizontal`,  
`ATS.Compute.Invert.PostSweep`

**Example**

```
Sub Main
    ATS.File.OpenTest "INVERT.AT2R"
    ATS.Compute.Invert.PostSweep = False
    ATS.Compute.Invert.Data(1) = True
    ATS.Compute.Invert.Horizontal("Hz") = 5000
    ATS.Sweep.Start
    ATS.Compute.Invert.Apply
End Sub
```

## ATS.Compute.Invert.Data

## Property

**Syntax**      `ATS.Compute.Invert.Data` (ByVal *Source* As Integer)

**Data Type**      Boolean

<i>True</i>	Select specified data.
<i>False</i>	Deselect specified data.

Parameter	Name	Description
	<i>Source</i>	1 = Data 1 measurements 2 = Data 2 measurements 3 = Data 3 measurements 4 = Data 4 measurements 5 = Data 5 measurements 6 = Data 6 measurements

**Description**      This command determines which data (1-6) the Invert computation is to be performed on. By using this command several times to select multiple data sources, several Invert computations can be performed in one operation.

**See Also**      `ATS.Compute.Invert.Apply`

**Example**      See example for `ATS.Compute.Invert.Apply`.

**ATS.Compute.Invert.Horizontal****Property**

<b>Syntax</b>	<code>ATS.Compute.Invert.Horizontal (ByVal Unit As String)</code>	
<b>Data Type</b>	Double	
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<code>Unit</code>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.
<b>Description</b>	This command sets the horizontal value in which the data will be inverted around.	
<b>See Also</b>	<code>ATS.Compute.Invert.Apply</code>	
<b>Example</b>	See example for <code>ATS.Compute.Invert.Apply</code> .	

**ATS.Compute.Invert.HorizontalUnit****Method**

<b>Syntax</b>	<code>ATS.Compute.Invert.HorizontalUnit</code>
<b>Result</b>	String
<b>Description</b>	This command returns the Unit used for the Horizontal setting for the Compute Invert function.
<b>See Also</b>	<code>ATS.Compute.Invert.Horizontal</code>
<b>Example</b>	<pre>Sub Main     ATS.Compute.Invert.Data(1) = True     ATS.Compute.Invert.Horizontal("Hz") = 1000.0     Debug.Print <b>ATS.Compute.Invert.HorizontalUnit</b> End Sub</pre>
<b>Output</b>	Hz

**ATS.Compute.Invert.PostSweep****Property**

<b>Syntax</b>	<code>ATS.Compute.Invert.PostSweep</code>
---------------	-------------------------------------------



<b>Data Type</b>	Boolean
	<i>True</i> Enable computation to be applied after sweep.
	<i>False</i> Disable computation after sweep.
<b>Description</b>	This command instructs the test to perform the Invert computation after a sweep is complete and sets the state of the Apply After Sweep field on the Compute Invert panel.
	ATS retains the order in which the Apply After Sweep field on the Compute panels is enabled. This permits multiple computations to be performed on data from a single test. The order of the computations is also retained in the test file.
<b>See Also</b>	ATS.Compute.Invert.Apply
<b>Example</b>	See example for ATS.Compute.Invert.Apply.

## ATS.Compute.Linearity.Apply

Method

<b>Syntax</b>	<b>ATS.Compute.Linearity.Apply</b>
<b>Result</b>	Boolean
	<i>True</i> Computation performed.
	<i>False</i> Computation NOT performed.
<b>Description</b>	This command applies the Linearity computation to the selected data (1-6). The difference
<b>See Also</b>	ATS.Compute.Clear.All, ATS.Compute.Linearity.Data, ATS.Compute.Linearity.PostSweep, ATS.Compute.Linearity.Start, ATS.Compute.Linearity.Stop,
<b>Example</b>	<pre> Sub Main   ATS.File.OpenTest "LINEAR.ATS2"   <b>ATS.Compute.Linearity.PostSweep</b> = False   <b>ATS.Compute.Linearity.Data</b>(1) = True   <b>ATS.Compute.Linearity.Start</b>("Vrms") = .5   <b>ATS.Compute.Linearity.Stop</b>("Vrms") = 2   ATS.Sweep.Start </pre>

```

ATS.Compute.Linearity.Apply
ATS.Sweep.Data(1).Bottom("V") = -.02
ATS.Sweep.Data(1).Top("V") = .02
End Sub

```

## ATS.Compute.Linearity.Data

## Property

**Syntax** `ATS.Compute.Linearity.Data (ByVal Source As Integer)`

**Data Type** Boolean

*True* Select specified data.  
*False* Deselect specified data.

Parameter	Name	Description
	<i>Source</i>	1 = Data 1 measurements 2 = Data 2 measurements 3 = Data 3 measurements 4 = Data 4 measurements 5 = Data 5 measurements 6 = Data 6 measurements

**Description** This command determines which data (1-6) the Linearity computation is to be performed on. By using this command several times to select multiple data sources, several Linearity computations can be performed in one operation.

**See Also** `ATS.Compute.Linearity.Apply`

**Example** See example for `ATS.Compute.Linearity.Apply`.

## ATS.Compute.Linearity.PostSweep

## Property

**Syntax** `ATS.Compute.Linearity.PostSweep`

**Data Type** Boolean

*True* Enable computation to be applied after sweep.  
*False* Disable computation after sweep.

**Description** This command instructs the test to perform the Linearity computation after a sweep is complete and sets the state of the Apply After Sweep field on the Compute Linearity panel.

ATS retains the order in which the Apply After Sweep field on the Compute panels is enabled. This permits multiple computations to be performed on data from a single test. The order of the computations is also retained in the test file.

**See Also** `ATS.Compute.Linearity.Apply`

**Example** See example for `ATS.Compute.Linearity.Apply`.

## ATS.Compute.Linearity.Start

**Property**

**Syntax** `ATS.Compute.Linearity.Start` (ByVal *Unit* As String)

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.

**Description** This command sets the Start value of the data over which the Linearity computation will be performed.

**See Also** `ATS.Compute.Linearity.Stop`,  
`ATS.Compute.Linearity.Apply`

**Example** See example for `ATS.Compute.Linearity.Apply`.

## ATS.Compute.Linearity.StartUnit

**Method**

**Syntax** `ATS.Compute.Linearity.StartUnit`

**Result** String

**Description** This command returns the Unit used for the Start setting for the Compute Linearity function.

**See Also** `ATS.Compute.Linearity.StopUnit`

**Example**

```
Sub Main
    ATS.Compute.Linearity.Data(1) = True
    ATS.Compute.Linearity.Start("Hz") = 0.02
    Debug.Print ATS.Compute.Linearity.StartUnit
End Sub
```

**Output** Hz

---

## ATS.Compute.Linearity.Stop

**Property**

**Syntax** `ATS.Compute.Linearity.Stop (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.

**Description** This command sets the Stop value of the data over which the Linearity computation will be performed.

**See Also** `ATS.Compute.Linearity.Start`,  
`ATS.Compute.Linearity.Apply`

**Example** See example for `ATS.Compute.Linearity.Apply`.

---

## ATS.Compute.Linearity.StopUnit

**Method**

**Syntax** `ATS.Compute.Linearity.StopUnit`

**Result** String

**Description** This command returns the Unit used for the Stop setting for the Compute Linearity function.

**See Also** `ATS.Compute.Linearity.StartUnit`

**Example**

```
Sub Main
    ATS.Compute.Linearity.Data(1) = True
    ATS.Compute.Linearity.Stop("Hz") = 20000.0
    Debug.Print ATS.Compute.Linearity.StopUnit
End Sub
```

**Output** Hz

---

## ATS.Compute.Max.Apply

## Method

**Syntax** `ATS.Compute.Max.Apply`

**Result** Boolean

*True* Computation performed.  
*False* Computation NOT performed.

**Description** This command applies the Maximum computation to the selected data (1-6).

**See Also** `ATS.Compute.Clear.All`, `ATS.Compute.Max.Data`,  
`ATS.Compute.Max.PostSweep`,  
`ATS.Compute.Max.Start`, `ATS.Compute.Max.Stop`

**Example**

```
Sub Main
    ATS.File.OpenTest "MAX.ATS2"
    ATS.Compute.Max.PostSweep = False
    ATS.Compute.Max.Data(1) = True
    ATS.Compute.Max.Start("Hz") = 2000
    ATS.Compute.Max.Stop("Hz") = 200
    ATS.Sweep.Start
    ATS.Compute.Max.Apply
End Sub
```

---

## ATS.Compute.Max.Data

## Property

**Syntax** `ATS.Compute.Max.Data (ByVal Source As Integer)`

**Data Type** Boolean

*True* Select specified data.  
*False* Deselect specified data.

Parameter	Name	Description
	<i>Source</i>	1 = Data 1 measurements 2 = Data 2 measurements 3 = Data 3 measurements 4 = Data 4 measurements 5 = Data 5 measurements 6 = Data 6 measurements

**Description** This command determines which data (1-6) the Maximum computation is to be performed on. By using this command several times to select multiple data sources, several Maximum computations can be performed in one operation.

**See Also** `ATS.Compute.Max.Apply`

**Example** See example for `ATS.Compute.Max.Apply`.

---

## ATS.Compute.Max.PostSweep

### Property

**Syntax** `ATS.Compute.Max.PostSweep`

**Data Type** Boolean

*True* Enable computation to be applied after sweep.  
*False* Disable computation after sweep.

**Description** This command instructs the test to perform the Maximum computation after a sweep is complete and sets the state of the Apply After Sweep field on the Compute Maximum panel.

ATS retains the order in which the Apply After Sweep field on the Compute panels is enabled. This permits multiple computations to be performed on data from a single test. The order of the computations is also retained in the test file.

**See Also** `ATS.Compute.Max.Apply`

**Example** See example for `ATS.Compute.Max.Apply`.

---

**ATS.Compute.Max.Start****Property****Syntax**      **ATS.Compute.Max.Start** (ByVal *Unit* As String)**Data Type**      Double

Parameter	Name	Description
	<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.

**Description**      This command sets the Start value of the data over which the Maximum computation will be performed.**See Also**      `ATS.Compute.Max.Stop`, `ATS.Compute.Max.Apply`**Example**      See example for `ATS.Compute.Max.Apply`.

---

**ATS.Compute.Max.StartUnit****Method****Syntax**      **ATS.Compute.Max.StartUnit****Result**      String**Description**      This command returns the Unit used for the Start setting for the Compute Maximum function.**See Also**      `ATS.Compute.Max.StopUnit`

**Example**

```
Sub Main
    ATS.Compute.Max.Data(1) = True
    ATS.Compute.Max.Start("Hz") = 0.02
    Debug.Print ATS.Compute.Max.StartUnit
End Sub
```

**Output**      Hz

---

**ATS.Compute.Max.Stop****Property****Syntax**      **ATS.Compute.Max.Stop** (ByVal *Unit* As String)

<b>Data Type</b>	Double				
<b>Parameter</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>Unit</i></td> <td>The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.</td> </tr> </tbody> </table>	Name	Description	<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.
Name	Description				
<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.				
<b>Description</b>	This command sets the Stop value of the data over which the Maximum computation will be performed.				
<b>See Also</b>	ATS.Compute.Max.Start, ATS.Compute.Max.Apply				
<b>Example</b>	See example for ATS.Compute.Max.Apply.				

---

## ATS.Compute.Max.StopUnit

Method

<b>Syntax</b>	<b>ATS.Compute.Max.StopUnit</b>
<b>Result</b>	String
<b>Description</b>	This command returns the Unit used for the Stop setting for the Compute Maximum function.
<b>See Also</b>	ATS.Compute.Max.StartUnit
<b>Example</b>	<pre>Sub Main     ATS.Compute.Max.Data(1) = True     ATS.Compute.Max.Stop("Hz") = 20000.0     Debug.Print <b>ATS.Compute.Max.StopUnit</b> End Sub</pre>
<b>Output</b>	Hz

---

## ATS.Compute.Min.Apply

Method

<b>Syntax</b>	<b>ATS.Compute.Min.Apply</b>				
<b>Result</b>	Boolean				
	<table> <tr> <td><i>True</i></td> <td>Computation performed.</td> </tr> <tr> <td><i>False</i></td> <td>Computation NOT performed.</td> </tr> </table>	<i>True</i>	Computation performed.	<i>False</i>	Computation NOT performed.
<i>True</i>	Computation performed.				
<i>False</i>	Computation NOT performed.				



**Description** This command applies the Minimum computation to the selected data (1-6).

**See Also** `ATS.Compute.Clear.All`, `ATS.Compute.Min.Data`, `ATS.Compute.Min.PostSweep`, `ATS.Compute.Min.Start`, `ATS.Compute.Min.Stop`,

**Example**

```
Sub Main
    ATS.File.OpenTest "MIN.ATS2"
    ATS.Compute.Min.PostSweep = False
    ATS.Compute.Min.Data(1) = True
    ATS.Compute.Min.Start("Hz") = 10000
    ATS.Compute.Min.Stop("Hz") = 200
    ATS.Sweep.Start
    ATS.Compute.Min.Apply
End Sub
```

---

## ATS.Compute.Min.Data

## Property

**Syntax** `ATS.Compute.Min.Data` (ByVal *Source* As Integer)

**Data Type** Boolean

<i>True</i>	Select specified data.
<i>False</i>	Deselect specified data.

Parameter	Name	Description
	<i>Source</i>	1 = Data 1 measurements 2 = Data 2 measurements 3 = Data 3 measurements 4 = Data 4 measurements 5 = Data 5 measurements 6 = Data 6 measurements

**Description** This command determines which data (1-6) the Minmum computation is to be performed on. By using this command several times to select multiple data sources, several Minmum computations can be performed in one operation.

**See Also** `ATS.Compute.Min.Apply`

**Example** See example for `ATS.Compute.Min.Apply`.

## ATS.Compute.Min.PostSweep

### Property

**Syntax** `ATS.Compute.Min.PostSweep`

**Data Type** Boolean

*True* Enable computation to be applied after sweep.  
*False* Disable computation after sweep.

**Description** This command instructs the test to perform the Minimum computation after a sweep is complete and sets the state of the Apply After Sweep field on the Compute Minimum panel.

ATS retains the order in which the Apply After Sweep field on the Compute panels is enabled. This permits multiple computations to be performed on data from a single test. The order of the computations is also retained in the test file.

**See Also** `ATS.Compute.Min.Apply`

**Example** See example for `ATS.Compute.Min.Apply`.

## ATS.Compute.Min.Start

### Property

**Syntax** `ATS.Compute.Min.Start (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.

**Description** This command sets the Start value of the data over which the Minimum computation will be performed.

**See Also** `ATS.Compute.Min.Stop`, `ATS.Compute.Min.Apply`

**Example** See example for `ATS.Compute.Min.Apply`.

---

## ATS.Compute.Min.StartUnit Method

<b>Syntax</b>	<code>ATS.Compute.Min.StartUnit</code>
<b>Result</b>	String
<b>Description</b>	This command returns the Unit used for the Start setting for the Compute Minimum function.
<b>See Also</b>	<code>ATS.Compute.Min.StopUnit</code>
<b>Example</b>	<pre>Sub Main     ATS.Compute.Min.Data(1) = True     ATS.Compute.Min.Start("Hz") = 0.02     Debug.Print <b>ATS.Compute.Min.StartUnit</b> End Sub</pre>
<b>Output</b>	Hz

---

## ATS.Compute.Min.Stop Property

<b>Syntax</b>	<code>ATS.Compute.Min.Stop (ByVal Unit As String)</code>	
<b>Data Type</b>	Double	
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.
<b>Description</b>	This command sets the Stop value of the data over which the Minimum computation will be performed.	
<b>See Also</b>	<code>ATS.Compute.Min.Start</code> , <code>ATS.Compute.Min.Apply</code>	
<b>Example</b>	See example for <code>ATS.Compute.Min.Apply</code> .	

---

## ATS.Compute.Min.StopUnit Method

<b>Syntax</b>	<code>ATS.Compute.Min.StopUnit</code>
---------------	---------------------------------------

<b>Result</b>	String
<b>Description</b>	This command returns the Unit used for the Stop setting for the Compute Minimum function.
<b>See Also</b>	ATS.Compute.Min.StartUnit
<b>Example</b>	<pre>Sub Main     ATS.Compute.Min.Data(1) = True     ATS.Compute.Min.Stop("Hz") = 20000.0     Debug.Print <b>ATS.Compute.Min.StopUnit</b> End Sub</pre>
<b>Output</b>	Hz

---

## ATS.Compute.Normalize.Apply

## Method

<b>Syntax</b>	<b>ATS.Compute.Normalize.Apply</b>
<b>Result</b>	Boolean
	<p><i>True</i>                      Computation performed.</p> <p><i>False</i>                     Computation NOT performed.</p>
<b>Description</b>	This command applies the Normalize computation to the selected data (1-6).
<b>See Also</b>	ATS.Compute.Clear.All, ATS.Compute.Normalize.Data, ATS.Compute.Normalize.Horizontal, ATS.Compute.Normalize.PostSweep, ATS.Compute.Normalize.Target
<b>Example</b>	<pre>Sub Main     ATS.File.OpenTest "NORMAL.ATS2"     <b>ATS.Compute.Normalize.PostSweep</b> = False     <b>ATS.Compute.Normalize.Data</b>(1) = True     <b>ATS.Compute.Normalize.Horizontal</b>("Hz") = 1000     <b>ATS.Compute.Normalize.Target</b>("dBV") = 0.0     ATS.Sweep.Start     <b>ATS.Compute.Normalize.Apply</b> End Sub</pre>

## ATS.Compute.Normalize.Data

## Property

**Syntax** `ATS.Compute.Normalize.Data (ByVal Source As Integer)`

**Data Type** Boolean

*True* Select specified data.  
*False* Deselect specified data.

**Parameter**

Name	Description
<i>Source</i>	1 = Data 1 measurements 2 = Data 2 measurements 3 = Data 3 measurements 4 = Data 4 measurements 5 = Data 5 measurements 6 = Data 6 measurements

**Description** This command determines which data (1-6) the Normalize computation is to be performed on. By using this command several times to select multiple data sources, several Normalize computations can be performed in one operation.

**See Also** `ATS.Compute.Normalize.Apply`

**Example** See example for `ATS.Compute.Normalize.Apply`.

## ATS.Compute.Normalize.Horizontal

## Property

**Syntax** `ATS.Compute.Normalize.Horizontal (ByVal Unit As String)`

**Data Type** Double

**Parameter**

Name	Description
<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.

**Description** This command sets the horizontal value in which the data will be normalized around.

**See Also** `ATS.Compute.Normalize.Apply`,  
`ATS.Compute.Normalize.Target`

**Example** See example for `ATS.Compute.Normalize.Apply`.

---

## ATS.Compute.Normalize.HorizontalUnit

**Method**

**Syntax** `ATS.Compute.Normalize.HorizontalUnit`

**Result** String

**Description** This command returns the Unit used for the Horizontal setting for the Compute Normalize function.

**See Also** `ATS.Compute.Normalize.Horizontal`

**Example**

```
Sub Main
    ATS.Compute.Normalize.Data(1) = True
    ATS.Compute.Normalize.Horizontal("Hz") = 1000.0
    Debug.Print ATS.Compute.Normalize.HorizontalUnit
End Sub
```

**Output** Hz

---

## ATS.Compute.Normalize.PostSweep

**Property**

**Syntax** `ATS.Compute.Normalize.PostSweep`

**Data Type** Boolean

*True* Enable computation to be applied after sweep.  
*False* Disable computation after sweep.

**Description** This command instructs the test to perform the Normalize computation after a sweep is complete and sets the state of the Apply After Sweep field on the Compute Normalize panel.

ATS retains the order in which the Apply After Sweep field on the Compute panels is enabled. This permits multiple computations to be performed on data from a single test. The order of the computations is also retained in the test file.

**See Also** `ATS.Compute.Normalize.Apply`

**Example** See example for `ATS.Compute.Normalize.Apply`.

---

## ATS.Compute.Normalize.Target

**Property**

**Syntax** `ATS.Compute.Normalize.Target (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	The desired unit has to be available to the sweep panel: Source 1 for X-Y plots and Data 2 for X-Y Data 2 On X plots.

**Description** This command sets the vertical value in which the data will be Normalized to.

**See Also** `ATS.Compute.Normalize.Apply`,  
`ATS.Compute.Normalize.Horizontal`

**Example** See example for `ATS.Compute.Normalize.Apply`.

---

## ATS.Compute.Normalize.TargetUnit

**Method**

**Syntax** `ATS.Compute.Normalize.TargetUnit`

**Result** String

**Description** This command returns the Unit used for the Target setting for the Compute Normalize function.

**See Also** `ATS.Compute.Normalize.Target`

**Example**

```
Sub Main
    ATS.Compute.Normalize.Data(1) = True
    ATS.Compute.Normalize.Target("V") = 1.0
    Debug.Print ATS.Compute.Normalize.TargetUnit
End Sub
```

**Output** v

---

## ATS.Compute.Smooth.Apply Method

**Syntax** `ATS.Compute.Smooth.Apply`

**Result** Boolean

*True* Computation performed.  
*False* Computation NOT performed.

**Description** This command performs a running 3-point smoothing computation to the selected data (1-6).

**See Also** `ATS.Compute.Clear.All`, `ATS.Compute.Smooth.Data`,  
`ATS.Compute.Smooth.Passes`,  
`ATS.Compute.Smooth.PostSweep`

**Example**

```
Sub Main
  ATS.File.OpenTest "SMOOTH.ATS2"
  ATS.Compute.Smooth.PostSweep = False
  ATS.Compute.Smooth.Auto = False
  ATS.Compute.Smooth.Data(1) = True
  ATS.Compute.Smooth.Passes = 1
  ATS.Sweep.Start
  ATS.Compute.Smooth.Apply
End Sub
```

---

## ATS.Compute.Smooth.Auto Property

**Syntax** `ATS.Compute.Smooth.Auto`

**Data Type** Boolean

*True* Enable auto smoothing.  
*False* Disable auto smoothing.

**Description** This command automatically determines the number of passes that the smoothing algorithm performs on the selected data based on the number of measurements in the data.



**See Also** `ATS.Compute.Smooth.Apply`

**Example** See example for `ATS.Compute.Smooth.Apply`.

---

## ATS.Compute.Smooth.Data

**Property**

**Syntax** `ATS.Compute.Smooth.Data (ByVal Source As Integer)`

**Data Type** Boolean

<i>True</i>	Select specified data.
<i>False</i>	Deselect specified data.

Parameter	Name	Description
	<i>Source</i>	1 = Data 1 measurements 2 = Data 2 measurements 3 = Data 3 measurements 4 = Data 4 measurements 5 = Data 5 measurements 6 = Data 6 measurements

**Description** This command determines which data (1-6) the Smooth computation is to be performed on. By using this command several times to select multiple data sources, several Smooth computations can be performed in one operation.

**See Also** `ATS.Compute.Smooth.Apply`

**Example** See example for `ATS.Compute.Smooth.Apply`.

---

## ATS.Compute.Smooth.Passes

**Property**

**Syntax** `ATS.Compute.Smooth.Passes`

**Data Type** Long

**Description** This command sets the number of times the smoothing algorithm is applied to the selected data.

**See Also** `ATS.Compute.Smooth.Apply`,  
`ATS.Compute.Smooth.Data`

**Example** See example for `ATS.Compute.Smooth.Apply`.

---

## ATS.Compute.Smooth.PostSweep

**Property**

**Syntax** `ATS.Compute.Smooth.PostSweep`

**Data Type** Boolean

*True* Enable computation to be applied after sweep.  
*False* Disable computation after sweep.

**Description** This command instructs the test to perform the Smooth computation after a sweep is complete and sets the state of the Apply After Sweep field on the Compute Smooth panel.

ATS retains the order in which the Apply After Sweep field on the Compute panels is enabled. This permits multiple computations to be performed on data from a single test. The order of the computations is also retained in the test file.

**See Also** `ATS.Compute.Smooth.Apply`

**Example** See example for `ATS.Compute.Smooth.Apply`.

---

## ATS.Compute.Status.Id

**Method**

**Syntax** `ATS.Compute.Status.Id(ByVal Num As Integer)`

**Data Type** Integer      Type of computation.

142	Normalize
138	Invert
144	Smooth
139	Linearity
136	Center
137	Delta
135	Average
140	Minimum
141	Maximum
151	Equalize

Parameter	Name	Description
	<i>Num</i>	Number representing computation order.
<b>Description</b>	This command returns the Compute Status Identification Number.	
<b>See Also</b>	ATS.Compute.Status.NumOf	
<b>Example</b>	<pre> Sub Main     ATS.File.OpenTest("Status Id.ats2")     Computations = <b>ATS.Compute.Status.NumOf</b>     Debug.Print Computations &amp; _         " Computations performed in the following order."      For Counter = 0 To Computations - 1         Debug.Print ComputationText _             (<b>ATS.Compute.Status.Id</b>(Counter))     Next Counter End Sub  Function ComputationText(IdNum)     Select Case (IdNum)         Case 142             ComputationText = "Normalize"         Case 138             ComputationText = "Invert"         Case 144             ComputationText = "Smooth"         Case 139             ComputationText = "Linearity"         Case 136             ComputationText = "Center"         Case 137             ComputationText = "Delta"         Case 135             ComputationText = "Average"         Case 140             ComputationText = "Minimum"         Case 141             ComputationText = "Maximum"         Case 151             ComputationText = "Equalize"     </pre>	

```

    End Select
End Function

```

**Output** 10 Computations performed in the following order

```

Normalize
Invert
Smooth
Linearity
Center
Delta
Average
Minimum
Maximum
Equalize

```

---

## ATS.Compute.Status.NumOf

## Method

**Syntax** `ATS . Compute . Status . NumOf`

**Data Type** Integer

**Description** This command returns the number of computations applied to the Sweep Data after the sweep has completed.

**See Also** `ATS . Compute . Status . Id`

**Example** See example for `ATS . Compute . Status . Id`.

### ATS.Data.AddRowToEnd

Method

**Syntax** `ATS.Data.AddRowToEnd (ByVal Id As Integer)`

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.

**Result** Integer

**Description** This command adds an additional row to the end of data and returns the number of the row added.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.Inst.Analyzer.FuncMode = apbAnlrPhase
    ATS.Sweep.Data(2).Id = 5905 'Add Phase to Sweep
    StartValue = 20
    StopValue = 20000
    Frequencies = 31

    Counter = StartValue
    Do
        intAddRowToEnd = ATS.Data.AddRowToEnd(0)
        ATS.Data.Value(0,0,intAddRowToEnd,"Hz") = _
            Counter
        ATS.Data.Value(0,1,intAddRowToEnd,"V") = 1.0
        ATS.Data.Value(0,2,intAddRowToEnd,"deg") = 0.0
        Counter = Counter * 1.25893
    Loop Until Counter > StopValue

    intAddRowToEnd = ATS.Data.AddRowToEnd(0)
    ATS.Data.Value(0,0,intAddRowToEnd,"Hz") = StopValue
```

```

ATS.Data.Value(0,1,intAddRowToEnd,"V") = 1.0
ATS.Data.Value(0,2,intAddRowToEnd,"deg") = 0.0
ATS.Data.UpdateDisplay(0)
End Sub

```

## ATS.Data.ColLimitError

## Method

**Syntax** `ATS.Data.ColLimitError (ByVal Id As Integer, ByVal Column As Integer)`

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
	<i>Column</i>	Number of the Data Column (0-7).

**Result** Integer

**Description** This command returns a positive value if any measurement exceeds the upper or lower limit values for the specified column of data. A zero is returned if no errors occur. The returned value defines the number of measurements that exceed a limit.

**See Also** `ATS.Data.LimitError`

**Example**

```

Sub Main
  ATS.File.OpenTest ("FREQ.ATS2")
  ATS.Sweep.Start
  Errors = ATS.Data.ColLimitError(0,1)
  If Errors > 0 Then
    ErrorsUpper = ATS.Data.ColUpperLimitError(0,1)
    ErrorsLower = ATS.Data.ColLowerLimitError(0,1)
    String1$ = "This test Failed. " & Str(Errors) _
      & " Errors."
    String2$ = Str(ErrorsUpper) & " Upper Limit _
      Errors."
    String3$ = Str(ErrorsLower) & " Lower Limit
      Errors."
    Prompt.Text = String1$ & Chr(13) & String2$ & _
      Chr(13) & String3$
    ATS.Prompt.FontSize = 18
  End If
End Sub

```

```

        ATS.Prompt.Position -1,-1,425,175
        ATS.Prompt.ShowWithContinue
        Stop
    ElseIf Errors = 0 Then
        ATS.Prompt.Text = "This test Passed."
        ATS.Prompt.FontSize = 18
        ATS.Prompt.Position -1,-1,290,100
        ATS.Prompt.ShowWithContinue
        Stop
    End If
End Sub

```

## ATS.Data.ColLowerLimitError

## Method

**Syntax** `ATS.Data.ColLowerLimitError (ByVal Id As Integer, ByVal Column As Integer)`

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
	<i>Column</i>	Number of the Data Column (0-7).

**Result** Integer

**Description** This command returns the number of lower limit errors for the selected data. A zero is returned if no errors occur.

**Example** See example for `ATS.Data.ColLimitError`.

## ATS.Data.ColName

## Method

**Syntax** `ATS.Data.ColName (ByVal Id As Integer, ByVal Column As Integer)`

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.

	<i>Column</i>	Number of the Data Column (0-7).
<b>Result</b>	String	
<b>Description</b>	This command returns the string as shown on the sweep panel for the selected data. The string defines the meter that is returning measurements.	
<b>Example</b>	<pre>Sub Main     ATS.Application.NewTest     ATS.Sweep.Start     ColumnName\$ = <b>ATS.Data.ColName</b>(0,1)     Debug.Print "String definition For Data 1: = " &amp; _     ColumnName\$ End Sub</pre>	
<b>Comment</b>	This macro puts up a prompt displaying the contents of the Data 1 control on the Sweep Panel.	
<b>Output</b>	String definition For Data 1: = .Analyzer.Level A	

---

## ATS.Data.ColNumOf

## Method

<b>Syntax</b>	<b>ATS.Data.ColNumOf</b> (ByVal <i>Id</i> As Integer)	
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
<b>Result</b>	Integer	
<b>Description</b>	This command returns the number of columns of data.	
<b>Example</b>	See example for <code>ATS.Data.ColSize</code> .	

---

## ATS.Data.ColSize

## Method

<b>Syntax</b>	<b>ATS.Data.ColSize</b> (ByVal <i>Id</i> As Integer, ByVal <i>Column</i> As Integer)	
---------------	--------------------------------------------------------------------------------------	--



Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
	<i>Column</i>	Number of the Data Column (0-7).
<b>Result</b>	Long	
<b>Description</b>	This command returns the number of rows in the specified column.	
<b>Example</b>	<pre>Sub Main     ATS.Application.NewTest     ATS2.AGen.OutputOn = True     ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon     ATS.Sweep.Start     NumColumns = <b>ATS.Data.ColNumOf</b>(0)     For Column = 1 To (NumColumns - 1) Step 1         Size = <b>ATS.Data.ColSize</b>(0, Column)         Debug.Print "Number of measurements for Column _             " &amp; Column &amp; " = " &amp; Size     Next Column End Sub</pre>	
<b>Output</b>	Number of measurements for Column 1 = 31	

## ATS.Data.ColUnit

## Property

<b>Syntax</b>	<b>ATS.Data.ColUnit</b> (ByVal <i>Id</i> As Integer, ByVal <i>Column</i> As Integer)	
Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
	<i>Column</i>	Number of the Data Column (0-7).
<b>Result</b>	String	
<b>Description</b>	This command returns the unit string for the data in the selected column. ATS computes all other relevant units for display.	
<b>Example</b>	<pre>Sub Main</pre>	

```

ATS.Application.NewTest
ATS2.AGen.OutputOn = True 'Turn Generator output ON
ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
ATS.Sweep.SinglePoint = True
ATS.Sweep.Start
UnitString = ATS.Data.ColUnit(0, 1)
Debug.Print "The Unit for Data 1 is " & UnitString
End Sub

```

## ATS.Data.ColUpperLimitError

Method

**Syntax**      **ATS.Data.ColUpperLimitError**(ByVal *Id* As Integer, ByVal *Column* As Integer)

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
	<i>Column</i>	Number of the Data Column (0-7).

**Result**      Integer

**Description**      This command returns the number of upper limit errors for the selected data column.

**Example**      See example for `ATS.Data.ColLimitError`.

## ATS.Data.DeleteRow

Method

**Syntax**      **ATS.Data.DeleteRow**(ByVal *Id* As Integer, ByVal *RowNum* As Integer)

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
	<i>RowNum</i>	Number of row to delete.

**Result**      Boolean

*True*      Row deleted.

*False*            Row not deleted.

**Description**    This command deletes the designated row.

Note that the row numbering begins with zero.

**Example**

```
Sub Main()
    Dim FreqData() As Double

    Size = ATS.Data.ColSize(0,0)
    ReDim FreqData(Size)
    FreqData = ATS.Data.Transfer(0,0, 0, Size - 1,"Hz")

    For Count1 = 0 To Size - 1
        LastDup = 0
        For Count2 = Count1 + 1 To Size - 1
            If FreqData(Count1) = FreqData(Count2) _
                And Count1 <> Count2 Then
                FreqData(Count2) = 0
                LastDup = Count2
                Data.Value(0,0,Count2,"Hz") = 0
            End If
        Next Count2
        If LastDup <> 0 Then Count1 = LastDup
    Next Count1

    Duplicates = 0
    For Count1 = Size - 1 To 0 Step -1
        If ATS.Data.Value(0,0,Count1,"Hz") = 0 Then
            ATS.Data.DeleteRow (0, count1)
            Duplicates = Duplicates + 1
        End If
    Next Count1
    If Duplicates > 0 Then
        ATS.Prompt.Text = Str$(Duplicates) & _
            " Duplicate frequency(s) removed."
        ATS.Prompt.Show
        Wait 2
        ATS.Prompt.Hide.
    End If
End Sub
```

**ATS.Data.Id****Method**

**Syntax** `ATS.Data.Id (ByVal FileName As String)`

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters. The file must be an ATS limit, sweep, or data (.adl, .ads, .atsa) file attached to the current test.

**Result** Integer

**Description** This command returns an ID# that identifies the file specified in the command argument. The ID# can be use as the *id%* argument in all of the Data commands to specify which data to act upon. Use an Id# of zero (0) to access sweep data.

**See Also** `ATS.Data.ColLimitError,`  
`ATS.Data.ColLowerLimitError,` `ATS.Data.ColName,`  
`ATS.Data.ColNumOf,` `ATS.Data.ColSize,`  
`ATS.Data.ColUpperLimitError,`  
`ATS.Data.LimitError,` `ATS.Data.LowerLimitError,`  
`ATS.Data.OptimizeDisplay,`  
`ATS.Data.UpdateDisplay,`  
`ATS.Data.UpperLimitError,` `ATS.Data.Value,`  
`ATS.Data.XferToArray`

**Example**

```
Sub Main
    Dim Limitarray As Variant
    Dim Tablearray As Variant
    File.OpenTest "ID.ATS2"
    LimitId = ATS.Data.Id("LIMIT.ATSL")
    TableId = ATS.Data.Id("TABLE.ATSS")
    LimitArray = ATS.Data.Transfer(LimitId, 1, 0, _
        ATS.Data.ColSize(LimitId,1) - 1, "V")
    TableArray = ATS.Data.Transfer(TableId, 0, 0, _
        ATS.Data.ColSize(TableId,0) - 1, "Hz")
    Debug.Print "Limit ID # = " & LimitId
    Debug.Print "Table ID # = " & TableId
End Sub
```

**Output**

```
Limit ID # = 100
Table ID # = 101
```

## ATS.Data.InsertRowAfter

## Method

**Syntax** `ATS.Data.InsertRowAfter (ByVal Id As Integer, ByVal RowNum As Integer)`

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
	<i>RowNum</i>	Number of row to insert after.

**Result** Boolean

<i>True</i>	Row inserted.
<i>False</i>	Row not inserted.

**Description** This command inserts an additional row before the designated row. Note that the row numbering begins with zero.

**Example** See example for `ATS.Data.InsertRowBefore`.

## ATS.Data.InsertRowBefore

## Method

**Syntax** `ATS.Data.InsertRowBefore (ByVal Id As Integer, ByVal RowNum As Integer)`

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
	<i>RowNum</i>	Number of row to insert before.

**Result** Boolean

<i>True</i>	Row deleted.
<i>False</i>	Row not deleted.

**Description** This command inserts an additional row after the designated row. Note that the row numbering begins with zero.

**Example** See example for `ATS.Data.InsertRowAfter`.

**ATS.Data.LimitError****Method**

**Syntax** `ATS.Data.LimitError (ByVal Id As Integer)`

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.

**Result** Integer

**Description** This command returns the number of measurements that exceed a limit.

**Example**

```

Sub Main
  ATS.File.OpenTest "S2-FREQ.ATS2"
  ATS.Sweep.Start
  Errors = ATS.Data.LimitError(0)
  If Errors > 0 Then
    ErrorsUpper = ATS.Data.ColUpperLimitError(0,1)
    ErrorsLower = ATS.Data.ColLowerLimitError(0,1)
    String1$ = "This test Failed. " & Str(Errors) _
      & " Errors. "
    String2$ = Str(ErrorsUpper) & " Upper Limit _
      Errors."
    String3$ = Str(ErrorsLower) & " Lower Limit _
      Errors."
    ATS.Prompt.Text = String1$ & vbCr & String2$ & _
      vbCr & String3$
    ATS.Prompt.FontSize = 18
    ATS.Prompt.Position -1,-1,425,175
    ATS.Prompt.ShowWithContinue
    Stop
  ElseIf Errors = 0 Then
    ATS.Prompt.Text = "This test Passed."
    ATS.Prompt.FontSize = 18
    ATS.Prompt.Position -1,-1,290,100
    ATS.Prompt.ShowWithContinue
    Stop
  End If
End Sub

```

## ATS.Data.LowerLimitError Method

**Syntax** `ATS.Data.LowerLimitError (ByVal Id As Integer)`

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.

**Result** Integer

**Description** This command returns a positive value if any measurement is less than the lower limit values. A zero is returned if no errors occur. The returned value defines the number of measurements that are less than the limit.

**Example**

```
Sub Main
  ATS.File.OpenTest "FREQ.ATS2"
  ATS.Sweep.Start
  Flag = ATS.Data.LowerLimitError(0)
  If Flag > 0 Then
    ATS.Prompt.Text = "This test Failed."
    ATS.Prompt.FontSize = 18
    ATS.Prompt.Position -1,-1,290,100
    ATS.Prompt.ShowWithContinue
    Stop
  ElseIf Flag = 0 Then
    ATS.Prompt.Text = "This test Passed."
    ATS.Prompt.FontSize = 16
    ATS.Prompt.Position -1,-1,290,100
    ATS.Prompt.ShowWithContinue
    Stop
  End If
End Sub
```

## ATS.Data.Status Property

**Syntax** `ATS.Data.Status (ByVal Id As Integer, ByVal Column As Integer, ByVal Index As Long, ByVal Status As Constant)`

<b>Data Type</b>	Boolean	
	<i>True</i>	Data value status as specified by Status constant.
	<i>False</i>	Data value status valid.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
	<i>Column</i>	Number of the Data Column (0-7).
	<i>Index</i>	This value defines which row a measurement is returned from. A column may have any number of rows. Use the <code>ATS.Data.ColSize</code> command to determine the number of rows in a column. Note that the row numbering begins with zero.
	<i>Status</i>	<code>apbInvalid</code> = Data displayed as Invalid. <code>apbTimeout</code> = Data displayed as Timed out. <code>apbUnregulated</code> = Data displayed as Unregulated.

**Description** This command set or returns the status of the specified data value.

**Example**

```

Sub Main
    Dim Timeouts As Integer
    Timeouts = 0
    ATS.File.OpenTest("Timeouts.ats2")
    For Row = 0 To (ATS.Data.ColSize(0, 1) - 1) Step 1
        Timeout = ATS.Data.Status(0, 1, Row, apbTimeout)
        If Timeout = True Then Timeouts = Timeouts + 1
    Next Row
    Debug.Print Timeouts & " timeouts detected."
End Sub

```

**Output** 2 timeouts detected.

---

## ATS.Data.Transfer

## Property

**Syntax** `ATS.Data.ArrayTransfer` (ByVal *Id* As Integer, ByVal *Column* As Integer, ByVal *StartIndex* As Long, ByVal *StopIndex* As Long, ByVal *Unit* As String)



**Data Type** Variant

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
	<i>Column</i>	Number of the Data Column (0-7).
	<i>StartIndex</i>	Starting row numbe.
	<i>StopIndex</i>	Ending row number.
	<i>Unit</i>	Refer to the setting or reading defined by the <i>column%</i> parameter to determine the appropriate unit selections.

**Description** This command transfers data to or from an array from memory. The order that data is placed into the array is defined by the *StartIndex* and *StopIndex*. Reversing the start and stop values will reverse the data taken from or placed into memory

**See Also** `ATS.Data.Id`

**Example**

```

Sub Main
    Dim A As Variant
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.Inst.Analyzer.FuncFilterLP = apbAnlrLP15k
    ATS.Sweep.Data(1).Id = 6343
    ATS.Sweep.Source(1).Start("Hz") = 100000
    ATS.Sweep.Source(1).Steps = 4
    ATS.Application.NewData
    ATS.Sweep.Start
    Size = ATS.Data.ColSize (0, 1) - 1
    A = ATS.Data.Transfer(0, 1, 0, 5, "V")
    For Counter = 0 To (Size - 1) Step 1
        Debug.Print "Reading" & Counter;" = " & _
            Format(A(Counter), "#.0000") & " V"
    Next Counter
End Sub
    
```

**Comment** The values in the Output are taken from the array and then displayed.

**Output** Reading 0 = .0223 V

```

Reading 1 = .9889 V
Reading 2 = .9953 V
Reading 3 = .9945 V
Reading4 = .9931 V

```

## ATS.Data.UpdateDisplay

## Method

**Syntax** `ATS.Data.UpdateDisplay (ByVal Id As Integer)`

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data only.

**Description** This command updates the data displayed in the table and graph displays.

### Example

```

Sub Main
    Dim A As Variant
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.Inst.Analyzer.FuncFilterLP = apbAnlrLP15k
    ATS.Sweep.Data(1).Id = 6343
    ATS.Sweep.Data(3).Id = 6343
    ATS.Sweep.Source(1).Start("Hz") = 30000
    ATS.Sweep.Source(1).Steps = 4
    ATS.Sweep.Start
    Size = ATS.Data.ColSize(0, 2) - 1
    A = ATS.Data.Transfer(0, 2, 0, Size, "V")

    For Reading = 0 To Size
        Debug.Print "Acquired Reading" & Reading & _
            " = " & Format(A(Reading), "#.0000") & " V"
        A(Reading) = A(Reading) * 1.20
        Debug.Print "Changed Reading" & Reading & _
            " = " & Format(A(Reading), "#.0000") & " V"
    Next Reading
    ATS.Data.UpdateDisplay(0)
End Sub

```

### Output

```
Acquired Reading 0 = .0219 V
```

```

Acquired Reading 1 = .9874 V
Acquired Reading 2 = .9937 V
Acquired Reading 3 = .9933 V
Acquired Reading 4 = .9950 V
Changed Reading 0 = .0262 V
Changed Reading 1 = 1.1848 V
Changed Reading 2 = 1.1925 V
Changed Reading 3 = 1.1920 V
Changed Reading 4 = 1.1940 V
    
```

## ATS.Data.UpperLimitError

## Method

**Syntax** `ATS.Data.UpperLimitError (ByVal Id As Integer)`

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.

**Result** Integer

**Description** This command returns a positive value if any measurement exceeds the upper limit values. A zero is returned if no errors occur. The returned value defines the number of measurements that exceed the limit.

**Example**

```

Sub Main
  ATS.File.OpenTest "FREQ.ATS2"
  ATS.Sweep.Start
  Flag = ATS.Data.UpperLimitError (0)
  If Flag > 0 Then
    ATS.Prompt.Text = "This test Failed."
    ATS.Prompt.FontSize = 18
    ATS.Prompt.Position -1,-1,290,100
    ATS.Prompt.ShowWithContinue
    Stop
  ElseIf Flag = 0 Then
    ATS.Prompt.Text = "This test Passed."
    ATS.Prompt.FontSize = 18
    ATS.Prompt.Position -1,-1,290,100
    ATS.Prompt.ShowWithContinue
    
```

```

        Stop
    End If
End Sub

```

## ATS.Data.Value

## Property

**Syntax**      **ATS.Data.Value** (ByVal *Id* As Integer, ByVal *Column* As Integer, ByVal *Index* As Long, ByVal *Unit* As String)

**Data Type**    Double

Parameter	Name	Description
	<i>Id</i>	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to <code>ATS.Data.Id</code> command for additional information.
	<i>Column</i>	Number of the Data Column (0-7).
	<i>Index</i>	This value defines which row a measurement is returned from. A column may have any number of rows. Use the <code>ATS.Data.ColSize</code> command to determine the number of rows in a column. Note that the row numbering begins with zero.
	<i>Unit</i>	Refer to the setting or reading defined by the <i>column%</i> parameter to determine the appropriate unit selections.

**Description**    This command returns the specified reading from sweep data.

**See Also**        `ATS.Data.ColSize`, `ATS.Data.Id`

**Example**

```

Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS.Sweep.SinglePoint = True
    ATS.Sweep.Start
    Reading1 = ATS.Data.Value(0, 1, 0, "V")
    Debug.Print "Reading = " & Format(Reading1, _
        "#.0000") & "V"
End Sub

```

**Output**         Reading = .9850 V

### ATSEvent\_OnAuxSetting

Event

**Syntax** `ATSEvent_OnAuxSetting (ByVal Id As Long, ByVal Value As Variant)`

Parameter	Name	Description
	<i>Id</i>	Setting identification 1 through 4.
	<i>Value</i>	Sweep source or settings bargraph control current value.

**Description** This event is called when a sweep source or settings bargraph control changes which in turn generates this event, for example, the Instrument parameter "Aux.Setting 1 (Double)" used as the Sweep panel source or Bargraph browser ID. Events are generated as the sweep runs or as the user manipulates the settings bargraph control.

#### Example

```
Public Halt As Boolean
Sub Main
    Halt = False

    ATS.Application.NewTest
    ATS2.AGen.ChBTrackA = False
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source (apbChA) = apbAnalogInGenMon
    ATS2.AnalogIn.Source (apbChB) = apbAnalogInGenMon

    BarId = ATS.BarGraph.New(6271)
    'Aux.Setting 1 (Double)
    ATS.BarGraph.AxisLeft (BarId, "") = 0.0
    ATS.BarGraph.AxisRight (BarId, "") = 2.0
    ATS.BarGraph.AxisIncrement (BarId, "") = 0.1

    BarId = ATS.BarGraph.New(6275)
    'Aux.Reading 1 (Double)
```

```

ATS.BarGraph.AxisLeft(BarId,"") = 0.2
ATS.BarGraph.AxisRight(BarId,"") = 4.0

ATS.Aux.Setting(1) = 1.0
ATS.Application.SetWatchDogTimer(1, 5.0, False)

Do
Loop While Halt = False

End Sub
Public Sub ATSEvent_OnAuxSetting(ByVal auxnum As
Long, ByVal Value As Double)
    if auxnum = 1 then
        ATS2.AGen.Ampl(apbChA, "Vrms") = Value - .2
        ATS2.AGen.Ampl(apbChB, "Vrms") = Value + .2
        ATS.Aux.Reading1Settling(auxnum, 3.0, 0.0, 1, _
            0.0, apbNone)
        ATS.Aux.ReadingTrig(auxnum)
        ReadyCount = ATS.Aux.ReadingReady(auxnum)
        ATS.Aux.SetReading(auxnum)
        ATS2.AGen.Ampl(apbChA, "Vrms") + _
            ATS2.AGen.Ampl(apbChB, "Vrms")
        ReadyCount = ATS.Aux.ReadingReady(auxnum)
    End If
End Sub
Public Sub ATSEvent_OnWatchDogTimeout(ByVal Id As _
Long)
    If Id = 1 Then
        Halt = True
    End If
End Sub

```

---

## ATSEvent\_OnDcxProgramControlInput

## Event

**Syntax**      **ATSEvent\_OnDcxProgramControlInput**(ByVal *Value* As Long)

Parameter	Name	Description
	<i>Value</i>	The value returned is the pin number of the DCX-127 Program Control Input connector pin that is pulled low. A zero (0) is returned if more than one button is pressed at a time.
<b>Description</b>	This event is called when one of the DCX-127 Program Control Input pins (1) is momentarily shorted to ground (Pin 9).	
<b>Example</b>	<pre>Dim Halt As Boolean Sub Main     Halt = False     Do         Loop While Halt = False     End Sub Sub <b>ATSEvent_OnDcxProgramControlInput</b> (ByVal Value As Long)     Debug.Print "Program Control = " &amp; Value     If Value = 0 Then Debug.Print "More than one button -     pressed."     If Value = 8 Then Halt = True End Sub</pre>	

## ATSEvent\_OnError

## Event

**Syntax** **ATSEvent\_OnError** (ByVal *Value* As Long)

Parameter	Name	Description
	<i>Value</i>	Error value as defined in Appendix D Error Codes

**Description** This event is called when an Error is encountered.

**Example**

```
Sub Main
    ATS2.AGen.Ampl (apbChA, "Vrms") = 111.9
    'Cause an error _
    and see what happens.
End Sub
Sub ATSEvent_OnError (ByVal ErrorCode As Long)
    Debug.Print "Got number " & Code & " " & _
    ATS.Application.GetCurrentErrorString
```

```

' If you are going to handle the error, then call
' ATS.Application.ClearCurrentError before you exit

' this subroutine to stop ATS from displaying the
' error,

ATS.Application.ClearCurrentError

' It is also preferable to call
' ATS.Application.ClearCurrentError before you
' make any other calls into ATS in case these
' calls also generate an unexpected error
End Sub

```

---

## ATSEvent\_OnSweepEnd

## Event

**Syntax**      **ATSEvent\_OnSweepEnd**

**Description**      This event is called when the sweep has terminated and the initial source value has been restored.

**Example**

```

Sub Main
    ATS.Sweep.SinglePoint = True
    ATS.Sweep.Start
End Sub
Sub ATSEvent_OnSweepStart()
    Debug.Print "Sweep Start"
End Sub
Sub ATSEvent_OnSweepNestStart(ByVal Source As Long)
    Debug.Print "Sweep Nest Start "
End Sub
Sub ATSEvent_OnSweepStep(ByVal Value As Variant, _
    Source As Long)
    Debug.Print "Sweep Step = " & Value
End Sub
Sub ATSEvent_OnSweepTrigger()
    Debug.Print "Sweep Trigger"
End Sub
Sub ATSEvent_OnSweepStepEnd()

```



```

        Debug.Print "Sweep Step End"
    End Sub
    Sub ATSEvent_OnSweepNestEnd()
        Debug.Print "Sweep Nest End"
    End Sub
    Sub ATSEvent_OnSweepEnd()
        Debug.Print "Sweep End"
    End Sub

```

**Output**

```

Sweep Start
Sweep Nest Start
Sweep Step = 20000
Sweep Trigger
Sweep Step End
Sweep Nest End
Sweep Step = 1000
Sweep End #

```

**ATSEvent\_OnSweepNestEnd****Event****Syntax**      **ATSEvent\_OnSweepNestEnd****Description**      This event is called after a single sweep is completed.**Example**      See example for ATSEvent\_OnSweepEnd.**ATSEvent\_OnSweepNestStart****Event****Syntax**      **ATSEvent\_OnSweepNestStart** (*ByVal Source As Long*)

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Source</i>	Sweep panel Source 1 Step value.

**Description**      This event is called before the first step of a sweep.**Example**      See example for ATSEvent\_OnSweepEnd.

---

## ATSEvent\_OnSweepStart Event

**Syntax**            `ATSEvent_OnSweepStart`

**Description**     This event is called at the start of a sweep. It prepares for the rest of the upcoming sweep by storing the initial sweep value and pre-calculating steps.

**See Also**         `ATSEvent_OnSweepEnd`

**Example**         See example for `ATSEvent_OnSweepEnd`.

---

## ATSEvent\_OnSweepStep Event

**Syntax**            `ATSEvent_OnSweepStep` (ByVal *Value* As Variant, ByVal *Source* as Long)

Parameter	Name	Description
	<i>Value</i>	Setting value.
	<i>Source</i>	Indicates Source 1 or Source 2 settings (1 or 2 only).

**Description**     This event is called after the setting for this sweep has been done.

**Example**         See example for `ATSEvent_OnSweepEnd`.

---

## ATSEvent\_OnSweepStepEnd Event

**Syntax**            `ATSEvent_OnSweepStepEnd`

**Description**     This event is called after a reading cycle has completed. The reading cycle may return up to six settled measurements.

**Example**         See example for `ATSEvent_OnSweepEnd`.

---

## ATSEvent\_OnSweepTrigger Event

**Syntax**            `ATSEvent_OnSweepTrigger`

**Description** This event is after a new step value is sent, to trigger a new reading cycle.

**Example** See example for ATSEvent\_OnSweepEnd.

---

## ATSEvent\_OnWatchDogTimeout

**Event**

**Syntax** `ATSEvent_OnWatchDogTimeout (ByVal Id As Long)`

Parameter	Name	Description
	<i>Id</i>	Timer identification 1 or 2.

**Description** This event is called when one of the two WatchDog Timers has expired.

**Example**

```

Dim Halt As Boolean
Sub Main
    Halt = False
    ATS.Application.NewTest
    ATS2.AGen.Output = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS.Sweep.Source(1).Steps = 30
    ATS.Application.SetWatchDogTimer(1, 5.0, False)
    ATS.Sweep.StartNoWait
    Do
        Loop While Halt = False
    End Sub
Sub ATSEvent_OnWatchDogTimeout(ByVal Id As Long)
    If Id = 1 Then
        Halt = True
        If ATS.Sweep.IsRunning = True Then
            ATS.Sweep.Stop
            Debug.Print "Sweep Stopped"
        End If
    End If
End Sub

```

User Notes

---

### ATS.File.AppendData

Method

**Syntax** `ATS.File.AppendData (ByVal FileName As String)`

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters.

<b>Result</b>	Boolean	
	<i>True</i>	File Data Append successfull.
	<i>False</i>	File Data Append failed.

**Description** This command appends data from the designated data file into memory. This comand will only load data from a data file that has identical Sweep panel Data 1-6 and Source 1-2 instrument parameters.

---

### ATS.File.ExportASCIIData

Method

**Syntax** `ATS.File.ExportASCIIData (ByVal FileName As String)`

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters.

<b>Result</b>	Boolean	
	<i>True</i>	File export successfull.
	<i>False</i>	File export failed.

**Description** This command saves the measurement data in memory to a coma delimited ASCII text file.

**See Also** `ATS.File.ImportASCIIData`

**Example**

```

Sub Main
    ATS.Application.NewTest
    ATS.File.ImportASCIIData ("TEMP.AT SX")
    ATS.Application.PanelOpen (apbGraph)
    ATS.Sweep.Data(1).LogLin = 1
    ATS.Graph.OptimizeLeft
    If ATS.Sweep.Data(1).Id <> 5049 Then _
        ATS.Compute.Smooth.Data(1) = True
    If ATS.Sweep.Data(2).Id <> 5049 Then _
        ATS.Compute.Smooth.Data(2) = True
    If ATS.Sweep.Data(3).Id <> 5049 Then _
        ATS.Compute.Smooth.Data(3) = True
    If ATS.Sweep.Data(4).Id <> 5049 Then _
        ATS.Compute.Smooth.Data(4) = True
    If ATS.Sweep.Data(5).Id <> 5049 Then _
        ATS.Compute.Smooth.Data(5) = True
    If ATS.Sweep.Data(6).Id <> 5049 Then _
        ATS.Compute.Smooth.Data(6) = True
    ATS.Compute.Smooth.Auto = True
    ATS.Compute.Smooth.Apply
    ATS.File.ExportASCIIData ("TEMP.AT SX")
End Sub

```

**ATS.File.ExportGraphic****Method**

**Syntax**      **ATS.File.ExportGraphic** (ByVal *FileName* As String, ByVal *Type* As Integer)

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters.
	<i>Type</i>	0 = Windows Meta File (.WMF). 1 = Windows Extended Meta File (.EMF).

**Result**      Boolean

<i>True</i>	File export successfull.
<i>False</i>	File export failed.

**Description**      This command saves the current graph measurement data in memory to the designated file.

**Example**

```

Sub Main
  On Error Resume Next
  ChDir MacroDir
  ATS.Application.NewTest
  ATS2.AGen.OutputOn = True
  ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
  ATS2.Inst.Analyzer.FuncFilterLP = apbAnlrLP15k
  ATS2.Inst.Analyzer.FuncFilterHP = apbAnlrHP400
  ATS.Sweep.Data(1).Id = 6343
  ATS.Sweep.Source(1).Start("Hz") = 30000
  ATS.Sweep.Start
  Kill "GRAPH.EMF"
  blnExport = ATS.File.ExportGraphic _
    ("GRAPH.EMF", 1)
  If blnExport = False Then End
  Dim MSWord As Object
  Set MSWord = CreateObject("Word.Basic")
  MSWord.AppShow
  MSWord.FileOpen Name:= MacroDir & "\GENERIC.DOC"
  MSWord.EditFind "Graph"
  MSWord.InsertPicture MacroDir & "\GRAPH.EMF"
  MSWord.FilePrint
  Wait 10
  MSWord.FileCloseAll 2
  MSWord.AppClose
End Sub

```

---

**ATS.File.ImportASCIIData**

**Method**

**Syntax**      **ATS.File.ImportASCIIData** (ByVal *FileName* As String)

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters.
<b>Result</b>	Boolean	
	<i>True</i>	File import successfull.
	<i>False</i>	File import failed.

**Description** This command loads into memory the designated ASCII data file. This comand only loads files that have been exported from ATS or conform to the ATS ASCII data file format.

**See Also** `ATS.File.ExportASCIIIData`

**Example** See example for `ATS.File.ExportASCIIIData`.

---

## ATS.File.ImportTest

**Method**

**Syntax** `ATS.File.ImportTest (ByVal FileName As String)`

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters. File (.at2c) must be from APWIN version 2.2x.

**Result** Boolean

<i>True</i>	File import successfull.
<i>False</i>	File import failed.

**Description** This command loads into memory the designated System Two Cascade test file. This comand only loads files that have been saved from APWIN versions 2.2x.

---

## ATS.File.OpenData

**Method**

**Syntax** `ATS.File.OpenData (ByVal FileName As String)`

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters.

**Result** Boolean

<i>True</i>	File open successfull.
<i>False</i>	File open failed.

**Description** This command loads the designated data file.

**Example**

```
Sub Main
    OpenResult = ATS.File.OpenTest("FRQ-RESP.ATS2")
    If OpenResult = False Then Call Open_Failed
```



```

OpenResult = ATS.File.OpenData ("FRQ-RESP.ATSA")
If OpenResult = False Then Call Open_Failed
ATS.Data.UpdateDisplay(0)
Wait 5
OpenResult = ATS.File.OpenTest("THD-FRQ.ATS2")
If OpenResult = False Then Call Open_Failed
OpenResult = ATS.File.OpenData ("THD-FRQ.ATSA")
If OpenResult = False Then Call Open_Failed
ATS.Data.UpdateDisplay(0)
Wait 5
OpenResult = ATS.File.OpenTest("RESIDNOI.ATS2")
If OpenResult = False Then Call Open_Failed
OpenResult = ATS.File.OpenData ("RESIDNOI.ATSA")
If OpenResult = False Then Call Open_Failed
ATS.Data.UpdateDisplay(0)
Wait 5
End Sub

Sub Open_Failed
    Debug.Print"File Open FAILED."
End
End Sub

```

## ATS.File.OpenMacro

## Method

**Syntax**      **ATS.File.OpenMacro** (ByVal *FileName* As String)

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters.
<b>Result</b>	Boolean	
	<i>True</i>	Not applicable.
	<i>False</i>	File open failed.

**Description**      This command loads the designated file into the macro editor and automatically runs the macro.

**Example**

```

Private Sub Form_Load()
    Dim ATS As Object
    Set ATS = CreateObject("ATS.Application")

```

```

ATS.Application.Visible = True
ATS.File.OpenMacro "C:\BUSY.ATSB"
While ATS.Macro.IsRunning = True
Wend
ChDir ATS.Application.MacroDir
ATS.Application.Quit
End Sub

```

---

## ATS.File.OpenTest

## Method

**Syntax**      **ATS.File.OpenTest** (ByVal *FileName* As String)

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters.
<b>Result</b>	Boolean	
	<i>True</i>	File open successfull.
	<i>False</i>	File open failed.

**Description**      This command loads the designated test file.

**Example**

```

Sub Main
  OpenResult = ATS.File.OpenTest ("FRQ-RESP.ATS2")
  If OpenResult = False Then Call Open_Failed
  ATS.Sweep.Start
  SaveResult = ATS.File.SaveDataAs ("FRQ-RESP.ATSA")
  If SaveResult = False Then Call Save_Failed

  OpenResult = ATS.File.OpenTest ("THD-FRQ.ATS2")
  If OpenResult = False Then Call Open_Failed
  ATS.Sweep.Start
  SaveResult = File.SaveDataAs ("THD-FRQ.ATSA")
  If SaveResult = False Then Call Save_Failed

  OpenResult = ATS.File.OpenTest ("RESIDNOI.ATS2")
  If OpenResult = False Then Call Open_Failed
  ATS.Sweep.Start
  SaveResult = ATS.File.SaveDataAs ("RESIDNOI.ATSA")
  If SaveResult = False Then Call Save_Failed

```

```

End Sub
Sub Open_Failed
    Debug.Print"Test Open FAILED."
End
End Sub
Sub Save_Failed
    Debug.Print"Test Save FAILED."
End
End Sub
    
```

## ATS.File.OpenWfm

## Method

**Syntax**      **ATS.File.OpenWfm**(ByVal *FileName* As String, ByVal *siOption1* As Integer, ByVal *siOption2* As Integer)

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters.
	<i>siOption1</i>	This option defines the buffer that the first waveform in a waveform file is loaded into. 0 = None 1 = Acquisition buffer 1 2 = Acquisition buffer 2 3 = Transform buffer 1 4 = Transform buffer 2
	<i>siOption2</i>	This option defines the buffer that the second waveform in a two-waveform file is loaded into. 0 = None 1 = Acquisition buffer 1 2 = Acquisition buffer 2 3 = Transform buffer 1 4 = Transform buffer 2

**Result**      Boolean

*True*              File open successfull.  
*False*             File open failed.

**Description**      This command loads the designated waveform file into the analyzer or generator buffers designated by Option1 and 2.

**Comments** Acquisition buffer : This buffer holds waveform data that has been generated by executing an acquisition (F9). Opening a waveform file containing a previously-acquired and saved waveform and specifying the acquisition buffer as the destination permits further analysis of the waveform including FFT spectrum analysis and waveform display.

Transform buffer : The transform buffer is the sub-section of the acquisition buffer starting at the FFT start time with a length equal to the presently-set FFT length.

Buffer 1 : This buffer is associated with the DSP channel 1.

Buffer 2 : This buffer is associated with the DSP channel 2.

Recommended file extensions :

Extension	Description
.AAM	Acquired waveform, 1 channel
.AAS	Acquired waveform, 2 channels

### Example

```
Sub Main
  ATS.File.OpenTest ("FFTSAVE.ATS2")
  OpenResult = ATS.File.OpenWfm("TEMP.AAS", 1, 2)
  If OpenResult = False Then Call Open_Failed
  ATS.Sweep.Reprocess
End Sub

Sub Open_Failed
  Debug.Print"Waveform Open FAILED."
End
End Sub
```

## ATS.File.SaveAll

## Method

**Syntax** **ATS.File.SaveAll**

**Description** This command saves the current test and all macros loaded in the macro editor.

**ATS.File.SaveDataAs****Method**

**Syntax**            **ATS.SaveDataAs** (ByVal *FileName* As String)

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters.
<b>Result</b>	Boolean	
	<i>True</i>	File save successfull.
	<i>False</i>	File save failed.

**Description**    This command saves the measurement data in memory to the designated file.

**Example**        See example for `ATS.File.OpenTest`.

**ATS.File.SaveTest****Method**

**Syntax**            **ATS.File.SaveTest**

<b>Result</b>	Boolean	
	<i>True</i>	File save successfull.
	<i>False</i>	File save failed.

**Description**    This command saves the current test.

**Example**

```

Sub Main
    ATS.File.OpenTest ("FRQ-RESP.AT2R")
    Sweep.Start
    If ATS.File.SaveTest = False Then GoTo Quit
    ATS.File.OpenTest ("THD-FRQ.AT2R")
    ATS.Sweep.Start
    If ATS.File.SaveTest = False Then GoTo Quit
    ATS.File.OpenTest ("RESIDNOI.AT2R")
    ATS.Sweep.Start
    If ATS.File.SaveTest = False Then GoTo Quit
    End
    Quit:
    Debug.Print "Test Save FAILED"
End Sub

```

**ATS.File.SaveTestAs****Method**

**Syntax** `ATS.File.SaveTestAs (ByVal FileName As String)`

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters.
<b>Result</b>	Boolean	
	<i>True</i>	File save successfull.
	<i>False</i>	File save failed.

**Description** This command saves the current test as defined by the panels to the designated file. The data currently in memory as well as panel and page configuration information is also saved in the test file.

**Example**

```
Sub Main
    OpenResult = ATS.File.OpenTest("FRQ-RESP.ATS2")
    If OpenResult = False Then Call Open_Failed
    ATS.Sweep.Start
    SaveResult = ATS.File.SaveTestAs("FRQ-RESP.ATS2")
    If SaveResult = False Then Call Save_Failed

    OpenResult = ATS.File.OpenTest("THD-FRQ.ATS2")
    If OpenResult = False Then Call Open_Failed
    ATS.Sweep.Start
    SaveResult = ATS.File.SaveTestAs("THD-FRQ.ATS2")
    If SaveResult = False Then Call Save_Failed

    OpenResult = ATS.File.OpenTest("RESIDNOI.ATS2")
    If OpenResult = False Then Call Open_Failed
    ATS.Sweep.Start
    SaveResult = ATS.File.SaveTestAs("RESIDNOI.ATS2")
    If SaveResult = False Then Call Save_Failed
End Sub

Sub Open_Failed
    Debug.Print"Test Open FAILED."
    End
End Sub

Sub Save_Failed
    Debug.Print"Test Save FAILED."
    End
```

End Sub

## ATS.File.SaveWfmAs

## Method

**Syntax**      **ATS.File.SaveWfmAs** (ByVal *FileName* As String, ByVal *siOption1* As Integer, ByVal *siOption2* As Integer)

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters.
	<i>siOption1</i>	This option determines the source of the waveform to be stored in the first section of the disk file. 1 = Acquisition buffer 1 2 = Acquisition buffer 2 3 = Transform buffer 1 4 = Transform buffer 2
	<i>siOption2</i>	This option determines the source of the waveform to be stored in the last section of the disk file. 0 = None 1 = Acquisition buffer 1 2 = Acquisition buffer 2 3 = Transform buffer 1 4 = Transform buffer 2

**Result**      Boolean

*True*              File save successfull.  
*False*             File save failed.

**Description**      This command saves waveform data contained in the buffers designated by Option #1 and #2 into the designated file. The waveform designated by Option #1 saves to the first section of the file and the Option #2 waveform to the last section of the file.

**Comment**          Acquisition buffer : This buffer holds waveform data captured into DSP memory by an acquisition (F9). Selecting the acquisition buffer causes the complete acquired signal to be saved to a disk file for later download (via the `ATS.File.OpenWfm` command) for further analysis including FFT spectrum analysis and waveform display.

Transform buffer : The transform buffer is the sub-section of the acquisition buffer starting at the FFT start time with a length equal to the presently-set FFT length. Selecting this option results in a smaller disk file since only a portion of the acquired signal is saved.

Buffer 1 : This buffer is associated to the DSP channel 1.

Buffer 2 : This buffer is associated to the DSP channel 2.

Recommended file extensions :

Extension	Description
.AAM	Acquired waveform, 1 channel
.AAS	Acquired waveform, 2 channels

### Example

```

Sub Main
  OpenResult = ATS.File.OpenTest "FFTSAVE.ATS2"
  If OpenResult = False Then Call Open_Failed
  ATS.Sweep.Start
  SaveResult = ATS.File.SaveWfmAs("TEMP.AAS", 1, 2)
  If SaveResult = False Then Call Save_Failed
End Sub
Sub Open_Failed
  Debug.Print "Open FAILED."
End
End Sub
Sub Save_Failed
  Debug.Print "Save FAILED."
End
End Sub

```



---

### ATS.Graph.Comment

Property

**Syntax**            `ATS.Graph.Comment`

**Data Type**        String            ASCII characters.

**Description**     This command transfers the ASCII characters to or from the comment section in the Graph panel to a string variable.

**See Also**         `ATS.Graph.CommentAppend`, `ATS.Graph.CommentShow`

**Example**            `Public Comment As String`

```
Sub Main
  ATS.Application.NewTest
  ATS2.AGen.OutputOn = True
  ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
  ATS.Application.Page = 2
  ATS.Graph.CommentShow = True
  Comment = "Run Test"

DisplayDialog:
  Begin Dialog UserDialog 170,84,.DlgHandler
    PushButton 20,14,130,21,"&Run Test",.PushButton1
    CancelButton 20,56,130,21
  End Dialog
  Dim dlg As UserDialog
  Select Case Dialog (dlg)
    Case 0
      ATS.Graph.CommentShow = False
    End
    Case 1
      ATS.Graph.Comment = "Test Running"
```

```

        Wait .5
        ATS.Sweep.Start
        Errors = ATS.Data.LimitError(0)
        If Errors >0 Then
            Comment = "Test FAILED"
        Else
            Comment = "Test PASSED"
        End If
    End Select
    GoTo DisplayDialog
End Sub

Private Function DlgHandler(DlgItem$, Action%,
    SuppValue%) As Boolean
    Select Case Action%
        Case 2
            DlgHandler = False
        Case 5
            DlgHandler = True
            ATS.Graph.Comment = Comment$
            Wait .5
            ATS.Graph.Comment = ""
    End Select
End Function

```

---

## ATS.Graph.CommentAppend

**Method**

**Syntax**      **ATS.Graph.CommentAppend** (ByVal *Text* As String)

**Data Type**    Void

**Parameter**    **Name**                      **Description**

*Text*                      ASCII text.

**Description**    This command appends the ASCII characters to the comment section in the Graph panel.

**See Also**        `ATS.Graph.Comment`, `ATS.Graph.CommentShow`

**ATS.Graph.CommentShow****Property**

<b>Syntax</b>	<code>ATS.Graph.CommentShow</code>
<b>Data Type</b>	Boolean
	<i>True</i> Display Comment section.
	<i>False</i> Remove Comment section from view.
<b>Description</b>	This command displays or removes from view the comment section in the Graph panel
<b>See Also</b>	<code>ATS.Graph.Comment</code> , <code>ATS.Graph.CommentAppend</code>
<b>Example</b>	See example for <code>ATS.Graph.Comment</code> .

**ATS.Graph.CopyToSweepPanel****Method**

<b>Syntax</b>	<code>ATS.Graph.CopyToSweepPanel</code>
<b>Description</b>	This command transfers the current graphic display vertical (Top/Bottom) and horizontal (Start/Stop) axis values to the Sweep panel Data 1, Data 2, and Sweep 1 settings.
<b>Example</b>	<pre> Sub Main     ATS.Application.NewTest     ATS2.AGen.OutputOn = True     ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon     ATS2.AnalogIn.Source(apbChB) = apbAnalogInGenMon     ATS2.Inst.Selection = apbInstFFTAnalyzer     ATS.Sweep.Data(1).Id = 6024     ATS.Sweep.Data(2).Id = 6027     ATS.Sweep.Source1.Id = 5515     ATS.Sweep.Start     <b>ATS.Graph.OptimizeLeft</b>     <b>ATS.Graph.CopyToSweepPanel</b>     ATS.Sweep.CopySettings(apbData1toData2)     Wait 5     <b>ATS.Graph.OptimizeRight</b>     <b>ATS.Graph.CopyToSweepPanel</b> </pre>

```

ATS.Sweep.CopySettings (apbData2toData1)
Wait 5
ATS.Graph.OptimizeTogether
Wait 5
ATS.Graph.OptimizeIndividually
End Sub

```

## ATS.Graph.CursorPosition

## Property

**Syntax** **ATS.Graph.CursorPosition** (ByVal *CursorNum* As Integer, ByVal *Unit* As String)

**Data Type** Double

Parameter	Name	Description
	<i>CursorNum</i>	1 = Cursor #1 2 = Cursor #2
	<i>Unit</i>	Refer to the setting or reading defined by the <i>column%</i> parameter to determine the appropriate unit selections.

**Description** This command returns the horizontal axis position value where the designated cursor is positioned.

**See Also** `ATS.Graph.CursorRow`

**Example**

```

Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source (apbChA) = apbAnalogInGenMon
    ATS2.AnalogIn.Source (apbChB) = apbAnalogInGenMon
    ATS2.Inst.Selection = apbInstFFTAnalyzer
    ATS2.Inst.FFT.Window = apbFFTNoneMoveToBinCenter
    ATS.Sweep.Data (1) .Id = 6024
    ATS.Sweep.Data (2) .Id = 6027
    ATS.Sweep.Source (1) .Id = 5515
    ATS.Sweep.CopySettings (apbData1ToData2)
    ATS.Sweep.Start
    ATS.Graph.OptimizeTogether
    ATS.Graph.CursorsOn (True)

```

```

ATS.Prompt.Text = "Select one of the Traces from _
    the Graph Legend then Position Cursor #1 on _
    the fundamental then press Continue."
ATS.Prompt.FontSize = 8
ATS.Prompt.Position(-1,-1,250,150)
ATS.Prompt.ShowWithContinue
Stop
Debug.Print "Frequency = " & _
    Format(ATS.Graph.CursorPosition(1, "Hz"), _
    "##.0000")
Debug.Print "Data Editor Row = " & _
    ATS.Graph.CursorRow(1)
Debug.Print "Value = " & _
    Format(ATS.Graph.CursorValue(1, "V"), "##.0000")
End Sub
    
```

## ATS.Graph.CursorRow

## Property

**Syntax**      **ATS.Graph.CursorRow** (ByVal *CursorNum* As Integer)

**Data Type**    Integer

Parameter	Name	Description
	<i>CursorNum</i>	1 = Cursor #1 2 = Cursor #2

**Description**    This command returns the nearest row number to the position of the designated cursor. The row number can be used to extract access data in with the `ATS.Data.Value` command.

**See Also**        `ATS.Graph.CursorPosition`

**Example**         See example for `ATS.Graph.CursorPosition`.

## ATS.Graph.CursorsOn

## Method

**Syntax**         **ATS.Graph.CursorsOn** (ByVal *bOn* As Boolean)

Parameter	Name	Description
	<i>bOn</i>	True = Display cursors.

		False = Remove cursors from view.
<b>Result</b>	Boolean	
	<i>True</i>	Cursors displayed.
	<i>False</i>	Cursors not displayed. This may be because the Graph Panel is not displayed.
<b>Description</b>	This command displays or removes from view the cursors on the Graph panel.	
<b>Example</b>	See example for <code>ATS.Graph.CursorPosition</code> .	

---

## ATS.Graph.CursorValue

## Property

<b>Syntax</b>	<code>ATS.Graph.CursorValue (ByVal CursorNum As Integer, ByVal Unit As String)</code>	
<b>Data Type</b>	Double	
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>CursorNum</i>	1 = Cursor #1 2 = Cursor #2
	<i>Unit</i>	Refer to the setting or reading defined by the <i>column%</i> parameter to determine the appropriate unit selections.
<b>Description</b>	This command returns the vertical axis value where the designated cursor is positioned.	
<b>See Also</b>	<code>ATS.Graph.CursorPosition</code>	
<b>Example</b>	See example for <code>ATS.Graph.CursorPosition</code> .	

---

## ATS.Graph.Label

## Property

<b>Syntax</b>	<code>ATS.Graph.Label (ByVal AxisId As Constant)</code>	
<b>Data Type</b>	String	ASCII text.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>AxisId</i>	apbAxisTop = Top center. apbAxisBottom = Bottom center.

apbAxisLeft = Left center.  
 apbAxisRight = Right center.

**Description** This command set or returns the graph axis Labels.

**Example**

```

Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS.Sweep.Start
    ATS.Graph.Title = "Title and Labels Example"
    ATS.Graph.LabelAuto(apbAxisLeft) = False
    ATS.Graph.Label(apbAxisLeft) = "Left"
    ATS.Graph.LabelAuto(apbAxisBottom) = False
    ATS.Graph.Label(apbAxisBottom) = "Bottom"
    ATS.Graph.LabelAuto(apbAxisRight) = False
    ATS.Graph.Label(apbAxisRight) = "Right"
    ATS.Graph.LabelAuto(apbAxisTop) = False
    ATS.Graph.Label(apbAxisTop) = "Top"
End Sub
    
```

---

## ATS.Graph.LabelAuto

## Property

**Syntax** **ATS.Graph.LabelAuto** (ByVal *AxisId* As Constant)

**Data Type** Boolean

<i>True</i>	Label text generated automatically based on the Sweep Panel settings.
<i>False</i>	Label defined programmatically.

Parameter	Name	Description
	<i>AxisId</i>	apbAxisTop = Top center. apbAxisBottom = Bottom center. apbAxisLeft = Left center. apbAxisRight = Right center.

**Description** This command specifies whether the graph labels are automatically generated based on the Sweep Panel settings or programmatically.

**Example** See example for `ATS.Graph.Label`.

---

**ATS.Graph.Legend.Comment****Property**

**Syntax**      **ATS.Graph.Legend.Comment** (ByVal SweepId As Integer, ByVal TraceId As Integer)

**Data Type**      String                  ASCII text.

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>SweepId</i>	Sweep number.
	<i>TraceId</i>	Trace number.

**Description**      This command transfers the ASCII characters to or from the Legend Trace comment section in the Graph panel to a string variable.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS.Sweep.Start
    ATS.Graph.Legend.LineColor(1, 1) = apbRed
    ATS.Graph.TraceShow(1, 1) = True
    ATS.Graph.Legend.LineThickness(1, 1) = 1
    ATS.Graph.Legend.LineStyle(1, 1) = apbSolid
    ATS.Graph.Legend.Comment(1, 1) = "Trace Comment."
End Sub
```

---

**ATS.Graph.Legend.LineColor****Property**

**Syntax**      **ATS.Graph.Legend.LineColor** (ByVal SweepId As Integer, ByVal TraceId As Integer)

**Data Type**      Constant  
*apbBlue*  
*apbCyan*  
*apbGray*  
*apbGreen*  
*apbMagenta*  
*apbRed*  
*apbYellow*



Parameter	Name	Description
	<i>SweepId</i>	Sweep number.
	<i>TraceId</i>	Trace number.
<b>Description</b>	This command sets the Legend Trace Color for the specified Sweep Trace.	
<b>Example</b>	See example for <code>ATS.Graph.Legend.Comment</code> .	

## ATS.Graph.Legend.LineStyle

### Property

**Syntax** `ATS.Graph.Legend.LineStyle (ByVal SweepId As Integer, ByVal TraceId As Integer)`

**Data Type** Constant  
*apbDash*  
*apbDashDot*  
*apbDashDotDot*  
*apbDot*  
*apbSolid*

Parameter	Name	Description
	<i>SweepId</i>	Sweep number.
	<i>TraceId</i>	Trace number.

**Description** This command sets the Legend Trace Line Style for the specified Sweep Trace.

**Example** See example for `ATS.Graph.Legend.Comment`.

## ATS.Graph.Legend.LineThickness

### Property

**Syntax** `ATS.Graph.Legend.LineThickness (ByVal SweepId As Integer, ByVal TraceId As Integer)`

**Data Type** Integer 1-32

Parameter	Name	Description
	<i>SweepId</i>	Sweep number.
	<i>TraceId</i>	Trace number.

**Description** This command sets the Line Thickness for the specified Sweep Trace.

**Example** **See example for** `ATS.Graph.Legend.Comment`.

---

## ATS.Graph.OptimizeIndividually

**Method**

**Syntax** `ATS.Graph.OptimizeIndividually`

**Description** This command optimizes the graph to display all data.

**See Also** `ATS.Graph.OptimizeLeft`, `ATS.Graph.OptimizeRight`,  
`ATS.Graph.OptimizeTogether`

**Example** See example for `ATS.Graph.CopyToSweepPanel`.

---

## ATS.Graph.OptimizeLeft

**Method**

**Syntax** `ATS.Graph.OptimizeLeft`

**Description** This command optimizes the graph to display the data on the Left axis (Data 1).

**See Also** `ATS.Graph.OptimizeIndividually`,  
`ATS.Graph.OptimizeRight`,  
`ATS.Graph.OptimizeTogether`

**Example** See example for `ATS.Graph.CopyToSweepPanel`.

---

## ATS.Graph.OptimizeRight

**Method**

**Syntax** `ATS.Graph.OptimizeRight`

**Description** This command optimizes the graph to display the data on the Right axis (Data 2).

**See Also** `ATS.Graph.OptimizeIndividually`,  
`ATS.Graph.OptimizeLeft`,  
`ATS.Graph.OptimizeTogether`

**Example** See example for `ATS.Graph.CopyToSweepPanel`.

## ATS.Graph.OptimizeTogether

**Method**

**Syntax** `ATS.Graph.OptimizeTogether`

**Description** This command optimizes the graph to display all data (both Data 1 and Data 2) using the same vertical axis values (Top and Bottom).

**See Also** `ATS.Graph.OptimizeIndividually`,  
`ATS.Graph.OptimizeLeft`, `ATS.Graph.OptimizeRight`

**Example** See example for `ATS.Graph.CopyToSweepPanel`.

## ATS.Graph.RefDataClear

**Method**

**Syntax** `ATS.Graph.OptimizeLeft`

**Description** This command Clears all Reference Data from memory. Clearing nonexistent Reference Data does not produce an error.

**See Also** `ATS.Graph.RefDataShow`, `ATS.Graph.RefDataStore`

**Example**

```

Sub Main
    ATS.Graph.RefDataClear
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon

    ATS.Sweep.Start
    ATS.Graph.Data(1).LogLin = apbLin
    ATS.Graph.RefDataStore
    ATS.Graph.RefDataShow = True
    ATS.Graph.OptimizeLeft
    ATS.Graph.CopyToSweepPanel
    ATS.Sweep.Start
End Sub
    
```

---

**ATS.Graph.RefDataShow****Property**

<b>Syntax</b>	<code>ATS.Graph.RefDataShow</code>	
<b>Data Type</b>	Boolean	
	<i>True</i>	Display Reference Data.
	<i>False</i>	Remove Reference Data from view.
<b>Description</b>	This command makes the Reference Data visible or invisible on the graph.	
<b>See Also</b>	<code>ATS.Graph.RefDataClear</code> , <code>ATS.Graph.RefDataStore</code>	
<b>Example</b>	See example for <code>ATS.Graph.RefDataClear</code> .	

---

**ATS.Graph.RefDataStore****Method**

<b>Syntax</b>	<code>ATS.Graph.RefDataStore</code>	
<b>Description</b>	This command adds the Sweep Data currently in memory to Reference Data memory.	
<b>See Also</b>	<code>ATS.Graph.RefDataClear</code> , <code>ATS.Graph.OptimizeRight</code>	
<b>Example</b>	See example for <code>ATS.Graph.RefDataClear</code> .	

---

**ATS.Graph.ScrollBarsOn****Property**

<b>Syntax</b>	<code>ATS.Graph.ScrollBarsOn</code>	
<b>Data Type</b>	Boolean	
	<i>True</i>	Display Scroll Bars.
	<i>False</i>	Remove Scroll Bars from view.
<b>Description</b>	This command makes the Scroll Bars visible or invisible on the graph.	

---

**ATS.Graph.Sweeps****Method**

<b>Syntax</b>	<code>ATS . Graph . Sweeps</code>
<b>Data Type</b>	Integer
<b>Description</b>	This command returns then number of Sweeps contained in the current data set.
<b>See Also</b>	<code>ATS . Graph . SweepTraces</code>

---

**ATS.Graph.SweepShow****Property**

<b>Syntax</b>	<code>ATS . Graph . SweepShow (ByVal SweepId As Integer)</code>	
<b>Data Type</b>	Boolean	
	<i>True</i>	Display Sweep Data.
	<i>False</i>	Remove Sweep Data from view.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>SweepId</i>	Sweep Data number.
<b>Description</b>	This command makes the specified Sweep set of traces visible or invisible on the graph.	
<b>See Also</b>	<code>ATS . Graph . Sweeps</code>	

---

**ATS.Graph.SweepTraces****Method**

<b>Syntax</b>	<code>ATS . Graph . SweepTraces</code>
<b>Data Type</b>	Integer
<b>Description</b>	This command returns then number of Sweeps contained in the current data set.
<b>See Also</b>	<code>ATS . Graph . Sweeps</code>

**ATS.Graph.TraceShow****Property**

**Syntax**      **ATS.Graph.TraceShow** (ByVal *SweepId* As Integer, ByVal *TraceId* As Integer)

**Data Type**      Constant

*apbBlue*  
*apbCyan*  
*apbGray*  
*apbGreen*  
*apbMagenta*  
*apbRed*  
*apbYellow*

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>SweepId</i>	Sweep number.
	<i>TraceId</i>	Trace number.

**Description**      This command makes the specified Trace visible or invisible on the graph.

**See Also**      `ATS.Graph.Sweeps`, `ATS.Graph.SweepTraces`

**ATS.Graph.ZoomOriginal****Method**

**Syntax**      **ATS.Graph.ZoomOriginal**

**Data Type**      Void

**Description**      This command replaces the current zoomed-in view with the graph coordinates in use when the most recent sweep was started, or with the default initial graph coordinates if no sweep has yet been made since ATS was launched.

**See Also**      `ATS.Graph.ZoomOut`

**ATS.Graph.ZoomOut****Method**

**Syntax**      **ATS.Graph.ZoomOut**

<b>Data Type</b>	Void
<b>Description</b>	This command causes the most recent zoom view to be replaced with the previous one. If you have zoomed repeatedly, the coordinates of each zoom have been saved in sequence in a memory stack. You may then Zoomout repeatedly to work back up through the stack, viewing the series of zoomed views in reverse order.
<b>See Also</b>	<code>ATS.Graph.ZoomOriginal</code>

User Notes



### ATS.LogFile.AddEntry Method

**Syntax** `ATS.LogFile.AddEntry (ByVal Text As String)`

Parameter	Name	Description
	<i>Text</i>	Any valid string

**Description** This command appends the current date and time and the defined string to the log file.

**Example**

```
Sub Main
    ATS.LogFile.Enable = True
    ATS.LogFile.FileName = "ATS-2.ALG"
    ATS.LogFile.ErrorMessages = True
    ATS.LogFile.FileActivity = True
    ATS.LogFile.PassFailMessages = True
    ATS.LogFile.TestName = True
    ATS.LogFile.GraphTitle = True
    ATS.LogFile.Data = 1
    ATS.LogFile.Clear
    ATS.LogFile.AddEntry "Test Start."
    ATS.LogFile.AddEntryWithoutTimeDate _
        "                               Amplitude Linearity."
    ATS.File.OpenTest "AMPLIN.ATS2"
    ATS.Sweep.Start
    ATS.LogFile.View
End Sub
```

**Output**

```
09/01/01 14:27:01    Test Start.
                    Amplitude Linearity.

09/01/01 14:27:01    Open Test: AMPLIN.ATS2
PASS 09/01/01 14:27:02    Execute sweep: AMPLIN.ATS2
```

**Comment**      The Example  
 Output is from the log file created when the example macro is run.

---

## ATS.LogFile.AddEntryWithoutTimeDate Method

**Syntax**      `ATS.LogFile.AddEntryWithoutTimeDate (ByVal Text  
 As String)`

Parameter	Name	Description
	<i>Text</i>	Any valid string

**Description**      This command appends the defined string to the log file.

**Example**      See example for `ATS.LogFile.AddEntry`.

---

## ATS.LogFile.Clear Method

**Syntax**      `ATS.LogFile.Clear`

**Description**      This command clears the contents of the current log file.

**Example**      See example for `ATS.LogFile.AddEntry`.

---

## ATS.LogFile.Data Property

**Syntax**      `ATS.LogFile.Data`

**Data Type**      Constant

<i>apbLogNone</i>	None
<i>apbLogAll</i>	All
<i>apbLogFailedDataOnly</i>	Failed data only

**Description**      This command controls whether no test point values (None), all test point values (All), or only those test points which were outside limits

(Failed Data Only) are written into the log file. Any values written into the log file which were outside limits will have parenthesis at the end with the (less than) or (greater than) symbol and the value of the limit which they failed.

**Example** See example for `ATS.LogFile.AddEntry`.

---

## ATS.LogFile.Enable

### Property

**Syntax** `ATS.LogFile.Enable`

**Data Type** Boolean  
*True* Enable  
*False* Disable

**Description** This command when enabled controls whether logging actually takes place. If disabled, no logging takes place

**Example** See example for `ATS.LogFile.AddEntry`.

---

## ATS.LogFile.ErrorMessages

### Property

**Syntax** `ATS.LogFile.ErrorMessages`

**Data Type** Boolean  
*True* Enable  
*False* Disable

**Description** This command when enabled logs into the log file any ATS or windows error messages which occur durring the period that logging is enabled.

**Example** See example for `ATS.LogFile.AddEntry`.

---

## ATS.LogFile.FileActivity

### Property

**Syntax** `ATS.LogFile.FileActivity`

**Data Type** Boolean

<i>True</i>	Enable
<i>False</i>	Disable

**Description** This command when enabled will enter into the log file a text message for every disk file opened or every file saved to disk. The message includes the name and full path name of the file, and the date and time at which it was opened or saved.

**Example** See example for `ATS.LogFile.AddEntry`.

## ATS.LogFile.FileName

**Property**

**Syntax** `ATS.LogFile.FileName`

**Data Type** String

**Description** This command defines the file name to use for the log file.

**Example** See example for `ATS.LogFile.AddEntry`.

## ATS.LogFile.GraphTitle

**Property**

**Syntax** `ATS.LogFile.GraphTitle`

**Data Type** Boolean

<i>True</i>	Enable
<i>False</i>	Disable

**Description** This command when enabled logs the Graph Title, and the Time and Date at which the test was executed, exactly as they are displayed in the title bar of the graph.

**Example** See example for `ATS.LogFile.AddEntry`.

## ATS.LogFile.PassFailMessages

**Property**

**Syntax** `ATS.LogFile.PassFailMessages`

**Data Type** Boolean

<i>True</i>	Enable
-------------	--------

*False*                  Disable

**Description**      This command when enabled causes an error summary message to be written into the log file each time a test is run. The first word of the message will be PASS or FAIL (See example for `ATS.LogFile.AddEntry`). Following a colon (:) the error message will include the number of measurements which where below the lower limit, the number of measurements that were above the upper limit, and the number of timeouts which occurred. If disabled, no error message is written to the error file.

**Example**              See example for `ATS.LogFile.AddEntry`.

## ATS.LogFile.PrintLogFile

## Method

**Syntax**                  **ATS.LogFile.PrintLogFile**

**Description**      This command loads the NOTEPAD application and prints the current log file and then closes the NOTEPAD application.

**Example**

```

Sub Main
  ATS.LogFile.Enable = True
  ATS.LogFile.FileName = "ATS-2.ALG"
  ATS.LogFile.ErrorMessages = True
  ATS.LogFile.FileActivity = True
  ATS.LogFile.PassFailMessages = True
  ATS.LogFile.TestName = True
  ATS.LogFile.GraphTitle = True
  ATS.LogFile.Data = apbLogAll
  ATS.LogFile.Clear
  ATS.LogFile.AddEntry _
    "                                Amplitude Linearity."
  ATS.File.OpenTest "AMPLIN.ATS2"
  ATS.Sweep.Start
  ATS.LogFile.PrintLogFile
End Sub

```

---

## ATS.LogFile.TestName

**Property**

<b>Syntax</b>	<code>ATS.LogFile.TestName</code>
<b>Data Type</b>	Boolean <i>True</i> Enable <i>False</i> Disable
<b>Description</b>	This command when enabled logs the test name, including path name, of the test when executed.
<b>Example</b>	See example for <code>ATS.LogFile.AddEntry</code> .

---

## ATS.LogFile.View

**Method**

<b>Syntax</b>	<code>ATS.LogFile.View</code>
<b>Description</b>	This command loads the NOTEPAD application and displays the current log file.
<b>Example</b>	See example for <code>ATS.LogFile.AddEntry</code> .

### ATS.Macro.IsRunning

OLE Method

**Syntax**            **ATS.Macro.IsRunning**

**Result**            Boolean

*True*                ATS Macro running.  
*False*               ATS Macro not running.

**Description**      This command returns the status of the ATS macro.

**Example**

```
Private Sub Form_Load()  
    Dim ATS As Object  
    Set ATS = CreateObject("ATS.Application")  
    ATS.Application.Visible = True  
  
    'Place your code here  
  
    ATS.File.OpenMacro "C:\BUSY.ATSB"  
    While ATS.Macro.IsRunning = True  
    Wend  
  
    ChDir ATS.Application.MacroDir  
  
    'Place your code here  
  
    ATS.Application.Quit  
End  
End Sub
```

---

**ATS.Macro.Name****Property**

**Syntax**            **ATS . Macro . Name**

**Result**            Boolean            ASCII characters.

**Description**      This command returns the ATS Macro Editor Name. This text string is located in the upper left corner of the ATS Macro/Procedure Editor before the Macro/Procedure name. This string is useful when using the AppActivate command in the Language reference section of Audio Precision Basic.

**See Also**            ATS.Application.Name

**Example**

```
Sub Main
  ATS.Application.VisibleMacroEditor = True
  AppActivate ATS.Application.Name
  SendKeys "%WC",1      'Clear all windows on page.
  SendKeys "%PO",1      'Display Data Editor.
  AppActivate ATS.Macro.Name

  'In Debug mode focus is automatically returned to
  ' the editor each time the user interacts with the
  ' controls. Therefore it is important to note that
  ' sections of code containing commands that are to
  ' be sent to other applications via the SendKeys
  ' command need to be executed without interruption.
  'When debugging these areas place breakpoints
  ' before and after the SendKeys commands to
  ' maintain the correct window/application focus.
End Sub
```



User Notes

User Notes

---

### ATS.Printout.Data

Method

**Syntax**            **ATS.Printout.Data**

**Result**            Boolean

*True*                Printout of tabular data successful.  
*False*              Printout of tabular data failed.

**Description**     This command prints the data displayed in the Data Editor in tabular format. The Data Editor is automatically displayed on the current page if it is not displayed on at least one of the five ATS Pages. The printer is defined by the settings on the File, Print Setup menu.

**See Also**         `ATS.Printout.Graph`

**Example**           `Sub Main`  
                      `ATS.File.OpenTest("GRAPH.ATS2")`  
                      `ATS.Application.PanelOpen(apbDataEditor)`  
                      `ATS.Sweep.Start`  
                      **`ATS.Printout.Data`**  
`End Sub`

---

### ATS.Printout.Graph

Method

**Syntax**            **ATS.Printout.Graph**

**Result**            Boolean

*True*                Printout of graphic data successful.  
*False*              Printout of graphic data failed.

**Description** This command sends the graph as defined by the settings on the File, Page Setup menu to the selected printer as define by the File, Print Setup menu. A graph must be displayed on at least one of the five ATS Pages to print or preview the graph.

**See Also** `ATS.Printout.LoadFromTest`

**Example**

```
Sub Main
  ATS.File.OpenTest ("GRAPH.ATS2")
  ATS.Printout.LoadFromTest
  ATS.File.OpenTest ("TEST.ATS2")
  ATS.Sweep.Start
  ATS.Printout.Graph
End Sub
```

---

## ATS.Printout.LoadFromTest

## Method

**Syntax** `ATS.Printout.LoadFromTest`

**Result** Boolean

*True* Page Setup settings loaded from test file.  
*False* Page Setup settings not loaded from test file.

**Description** This command loads the page setup settings from the currently loaded test. The printout settings are global and can only be changed via this command or manually via the user interacting with the Page Setup menu. This allows the system to produce graphic printouts that have a consistent format over many different tests. Loading a test doesn't automatically update the Page Setup menu and change the graphic output.

**See Also** `ATS.Printout.Graph`

**Example** See example macro `ATS.Printout.Graph`.

---

## ATS.Printout.TrackGraph

## Property

**Syntax** `ATS.Printout.TrackGraph`

<b>Data Type</b>	Boolean <i>True</i> Use Graph Trace settings. <i>False</i> Use settings defined on Page Setup menu.
<b>Description</b>	This command causes the printout Color, Line Style, and Thickness to automatically track the settings used in the Graph Window legend. A graph must be displayed on at least one of the five ATS Pages to print or preview the graph.
<b>See Also</b>	ATS.Printout.LoadFromTest
<b>Example</b>	<pre>Sub Main   ATS.Application.NewTest   <b>ATS.Printout.TrackGraph</b> = True   ATS2.AGen.OutputOn = True   ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon   ATS2.Inst.Analyzer.FuncFilterHP = apbAnlrHP400   ATS.Sweep.Start   ATS.Printout.Graph End Sub</pre>

## User Notes

---

### ATS.Prompt.FontSize

Property

**Syntax**            **ATS.Prompt.FontSize**

**Data Type**        Double            Default size = 16.

**Description**     This command sets the font size of the characters used in the User Prompt.

**Example**            Sub Main  
                      With **ATS.Prompt**  
                          **.Title** = "ATS Prompt"  
                          **.Text** = "This prompt will be shown \_  
                                      for 5 seconds."  
                          **.FontSize** = 8  
                          **.Position**(-1,-1,190,120)  
                          **.Show**  
                          Shown = **.IsUp**  
                          Wait 5  
                          **.Hide**  
                          Shown = **Prompt.IsUp**  
                          **.Text** = "This prompt will be shown until the \_  
                                      Continue Macro button is selected below."  
                          **.ShowWithContinue**  
                          Stop  
                          End With  
                      End Sub

---

### ATS.Prompt.Hide

Method

**Syntax**            **ATS.Prompt.Hide**

**Description** This command removes the user prompt from view.

**Example** See example macro `ATS.Prompt.FontSize`.

---

## ATS.Prompt.IsUp

**Method**

**Syntax** `ATS.Prompt.IsUp`

**Result** Boolean

*True* User prompt is displayed.

*False* User prompt is NOT displayed.

**Description** This command returns the current status of the user prompt.

**Example** See example for `ATS.Prompt.FontSize`.

---

## ATS.Prompt.Position

**Method**

**Syntax** `ATS.Prompt.Position(ByVal X As Integer, ByVal Y As Integer, ByVal CX As Integer, ByVal CY As Integer)`

**Parameter**

Name	Description
<i>X</i>	This number value is the distance from the left edge of the monitor screen. It is measured in 1/8ths of the average character width for the dialog's font. Setting this number value to -1 centers the prompt horizontally.
<i>Y</i>	This number value is the distance from the top edge of the monitor screen. It is measured in 1/12ths of the character height for the dialog's font. Setting this number value to -1 centers the prompt vertically.
<i>CX</i>	This number value is the width. It is measured in 1/8ths of the average character width for the dialog's font.
<i>CY</i>	This number value is the height. It is measured in 1/12ths of the character height for the dialog's font.

**Description** This command defines the position and size of the User Prompt.

**Example** See example for `ATS.Prompt.FontSize`.



---

## ATS.Prompt.Show

**Method**

- Syntax**            `ATS.Prompt.Show`
- Description**      This command displays the user prompt.
- See Also**            (Language Reference) Stop Instruction
- Example**            See example for `ATS.Prompt.FontSize`.

---

## ATS.Prompt.ShowWithContinue

**Method**

- Syntax**            `ATS.Prompt.ShowWithContinue`
- Description**      This command displays the current user prompt window with a continue button.
- See Also**            (Language Reference) Stop Instruction
- Example**            See example for `ATS.Prompt.FontSize`.

---

## ATS.Prompt.ShowWithContinueAndStopSweep

**Method**

- Syntax**            `ATS.Prompt.ShowWithContinueAndStopSweep`
- Description**      This command displays the current user prompt window with a continue button. When the continue button is selected a sweep if running is terminated.
- See Also**            (Language Reference) Stop Instruction
- Example**            Sub Main  
                       `ATS.Application.NewTest`  
                       `ATS2.AGen.Ampl(apbChA, "Vrms") = 2`  
                       `ATS2.AGen.OutputOn = True`  
                       `ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon`  
                       `ATS2.Inst.Analyzer.FuncMode = apbAnlrAmplitude`  
                       With **ATS.Prompt**  
                           `.Text = "Press the Continue button to STOP _`  
                           `the SWEEP and continue the Macro."`

```

        .FontSize = 8
        .Position (-1,-1,180,120)
        .ShowWithContinueAndStopSweep
ATS.Sweep.Repeat = True
ATS.Sweep.Start
        .Text = "Press the Continue button to _
            END the Macro."
        .FontSize = 8
        .Position (-1,-1,200,100)
        .ShowWithContinue
    Stop
End With
End Sub

```

---

## ATS.Prompt.Text

## Property

<b>Syntax</b>	<b>ATS.Prompt.Text</b>
<b>Data Type</b>	String            User defined string.
<b>Description</b>	This command defines the string to be displayed in a user prompt.
<b>Example</b>	See example for <code>ATS.Prompt.FontSize</code> .

---

## ATS.Prompt.Title

## Property

<b>Syntax</b>	<b>ATS.Prompt.Title</b>
<b>Data Type</b>	String            User defined string.
<b>Description</b>	This command transfers ASCII characters to or from the Title section in a Prompt panel to a string variable.
<b>Example</b>	See example for <code>ATS.Prompt.FontSize</code> .

# Chapter 16

## Regulation

### ATS.Reg.IsRunning

Method

**Syntax**            `ATS.Reg.IsRunning`

**Result**            Boolean

*True*                Regulation process running.  
*False*               Regulation process not running.

**Description**      This command returns the status of the Regulation process.

**See Also**           `ATS.Macro.LoadFromTest`

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.Ampl (apbChA, "dBV") = 0.0
    ATS.Application.PanelOpen (apbRegulation)
    ATS.Reg.TargetID = 6343
    ATS.Reg.TargetValue ("dBV") = -80.0
    ATS.Reg.TargetToleranceMode = apbRegdB
    ATS.Reg.TargetTolerance ("dB") = 1.0
    ATS.Reg.SourceID = 5052
    ATS.Reg.SourceHigh ("dBV") = -70.0
    ATS.Reg.SourceLow ("dBV") = -90.0
    ATS.Reg.SourceOperation = apbRegMinusNorm
    ATS.Reg.SourceStepSize ("dB") = 0.1
    ATS.Reg.SourceIteration = 30

    ATS.Reg.StartNoWait (True)
    StartTime = Timer
    Do
        Wait .1
        Debug.Print Timer
```

```

Loop While ATS.Reg.IsRunning And _
    Timer < StartTime + Reg.Timeout
If ATS.Reg.IsRunning = True Then
    ATS.Reg.StartNowait(False)
    Debug.Print "Regulation Stopped!"
End If
End Sub

```

---

## ATS.Reg.SourceHigh

## Property

<b>Syntax</b>	<b>ATS.Reg.SourceHigh</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	Refer to the setting defined by the <code>ATS.Reg.SourceId</code> command (ID#) to determine the appropriate range of acceptable values.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	Refer to the setting defined by the <code>ATS.RegSourceId</code> command (ID#) to determine the appropriate unit selections.
<b>Description</b>	This command sets the upper boundary for the source parameter used in the regulation process.	
<b>See Also</b>	<code>ATS.Reg.SourceLow</code>	
<b>Example</b>	See example for <code>ATS.Reg.SourceId</code> .	

---

## ATS.Reg.SourceId

## Property

<b>Syntax</b>	<b>ATS.Reg.SourceId</b>	
<b>Data Type</b>	Long	Instrument Parameter ID#.
<b>Description</b>	This command is used to select the instrument parameter which will define settings used in the regulation process.	
	Refer to Appendix B to obtain instrument parameter identification numbers.	
<b>Example</b>	Sub Main	

```

ATS.File.OpenTest "REG3.AT2R"
ATS.Reg.SourceID = 5051
ATS.Reg.TargetID = 6343
ATS.Reg.SourceOperation = apbRegPlusNorm
ATS.Reg.SourceStepSize("%") = 2
ATS.Reg.SourceIteration = 30
ATS.Reg.TargetToleranceMode = apbRegPercent
ATS.Reg.TargetValue("dBrA") = -3
ATS.Reg.TargetTolerance("%") = 5
ATS.Reg.SourceHigh("Hz") = 5000
ATS.Reg.SourceLow("Hz") = 200
ATS.Reg.SweepEnable = False
ATS.Reg.Timeout = 2.0
ATS.Reg.Start
End Sub
    
```

---

## ATS.Reg.Sourcelteration

## Property

<b>Syntax</b>	<code>ATS.Reg.SourceIteration</code>
<b>Data Type</b>	Long
<b>Description</b>	This command sets the number of Source Iterations. Iterations limit the maximum number of regulation attempt steps the source will make before exiting the search and moving on.
<b>See Also</b>	<code>ATS.Reg.SourceOperation</code>
<b>Example</b>	See example for <code>ATS.Reg.SourceId</code> .

---

## ATS.Reg.SourceLow

## Property

<b>Syntax</b>	<code>ATS.Reg.SourceLow (ByVal Unit As String)</code>
<b>Data Type</b>	Double
	Refer to the setting defined by the <code>ATS.Reg.SourceId</code> command (ID#) to determine the appropriate range of acceptable values.

Parameter	Name	Description
	<i>Unit</i>	Refer to the setting defined by the <code>ATS.RegSourceId</code> command (ID#) to determine the appropriate unit selections.
<b>Description</b>		This command sets the lower boundary for the source parameter used in the regulation process.
<b>See Also</b>	<code>ATS.Reg.SourceHigh</code>	
<b>Example</b>		See example for <code>ATS.Reg.SourceId</code> .

## ATS.Reg.SourceOperation

## Property

**Syntax** `ATS.Reg.SourceOperation`

**Data Type** Constant

*apbRegLinear*

Linear: assumes a linear relationship between the source setting and the target reading.

*apbRegPlusNorm*

+ Normal: assumes that an increasing source setting will cause an increasing target reading, but not necessarily linearly.

*apbRegMinusNorm*

- Normal: assumes that an increasing source setting will cause a decreasing target reading, but not necessarily linearly.

*apbRegMax*

Maximum: each cycle of regulation starts from the source low boundary value. For example the source will increase by the specified step size as long as the target value also increases. If the target value goes through a peak and starts to decrease, the direction of the source reverses and the step size is cut in half. These half-size steps continue until the target value again starts to decrease, at which time the direction of change again reverses and the step size is

again cut in half. This process will continue until the number of peak crossings equal the value defined by the `ATS.Reg.SourceIterations` command.

*apbRegMin*

Minimum: each cycle of regulation starts from the source low boundry value. For example the source will increase by the specified step size as long as the target value decreases. If the target value goes through a notch and starts to increase, the direction of the source reverses and the step size is cut in half. These half-size steps continue untill the target value again starts to increase, at which time the direction of change again reverses and the step size is again cut in half. This process will continue until the number of peak crossings equal the value defined by the `ATS.Reg.SourceIterations` command.

- Description** This command selects the type of algorithm used to control the source parameter specified by the `ATS.Reg.SourceId` command.
- See Also** `ATS.Reg.SourceStepSize`, `ATS.Reg.SourceIteration`
- Example** See example for `ATS.Reg.SourceId`.

## ATS.Reg.SourceStepSize

## Property

**Syntax** `ATS.Reg.SourceStepSize (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: X/Y, dB, %

**Description** This command sets the Source Step Size for the + Normal, - Normal, Maximum, and Minimum algorithm selections as it begins its search at the beginning of each new regulation process.

**See Also** `ATS.Reg.SourceOperation`, `ATS.Reg.SourceIteration`

**Example** See example for `ATS.Reg.SourceId`.

**ATS.Reg.Start****Method**

- Syntax**            `ATS.Reg.Start`
- Description**      This command initiates a regulation process.
- Example**            See example for `ATS.Reg.SourceId`.

**ATS.Reg.StartNoWait****Method**

- Syntax**            `ATS.Reg.StartNoWait (ByVal bStart As Boolean)`
- | Parameter | Name          | Description                                                          |
|-----------|---------------|----------------------------------------------------------------------|
|           | <i>bStart</i> | True = Start regulation process.<br>False = Stop regulation process. |
- Description**      This command initiates a regulation process and then continues macro execution.
- Example**            See example for `ATS.Reg.SourceId`.

**ATS.Reg.SweepEnable****Property**

- Syntax**            `ATS.Reg.SweepEnable`
- Data Type**        Boolean
- |              |                                         |
|--------------|-----------------------------------------|
| <i>True</i>  | Enable regulation for each sweep step.  |
| <i>False</i> | Disable regulation for each sweep step. |
- Description**      This command enables or disables regulation before each step in a sweep.
- See Also**          `ATS.Reg.Start`
- Example**            See example for `ATS.Reg.SourceId`.



**ATS.Reg.TargetId****Property**

<b>Syntax</b>	<code>ATS.Reg.TargetId</code>	
<b>Data Type</b>	Long	Instrument Parameter ID#.
<b>Description</b>	<p>This command is used to select the instrument parameter which will return readings used in the regulation process.</p> <p>Refer to Appendix B to obtain instrument parameter identification numbers.</p>	
<b>Example</b>	See example for <code>ATS.Reg.SourceId</code> .	

**ATS.Reg.TargetTolerance****Property**

<b>Syntax</b>	<code>ATS.Reg.TargetTolerance (ByVal Unit As String)</code>					
<b>Data Type</b>	Double	Refer to the setting defined by the <code>ATS.Reg.TargetId</code> command (ID#) to determine the appropriate range of acceptable values.				
<b>Parameter</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>Unit</i></td> <td>Refer to the setting defined by the <code>ATS.RegTargetId</code> command (ID#) to determine the appropriate unit selections for the Abs selection for the <code>ATS.Reg.TargetToleranceMode</code> command. The following units are available when % is selected with the <code>ATS.Reg.TargetToleranceMode</code> command: X/Y, %</td> </tr> </tbody> </table>	Name	Description	<i>Unit</i>	Refer to the setting defined by the <code>ATS.RegTargetId</code> command (ID#) to determine the appropriate unit selections for the Abs selection for the <code>ATS.Reg.TargetToleranceMode</code> command. The following units are available when % is selected with the <code>ATS.Reg.TargetToleranceMode</code> command: X/Y, %	
Name	Description					
<i>Unit</i>	Refer to the setting defined by the <code>ATS.RegTargetId</code> command (ID#) to determine the appropriate unit selections for the Abs selection for the <code>ATS.Reg.TargetToleranceMode</code> command. The following units are available when % is selected with the <code>ATS.Reg.TargetToleranceMode</code> command: X/Y, %					
<b>Description</b>	This command sets the tolerance that the regulation algorithm uses to determine if the regulation process is complete.					
<b>See Also</b>	<code>ATS.Reg.TargetToleranceMode</code> , <code>ATS.Reg.TargetValue</code>					
<b>Example</b>	See example for <code>ATS.Reg.SourceId</code> .					

**ATS.Reg.TargetToleranceMode****Property****Syntax** `ATS.Reg.TargetToleranceMode`

**Data Type** Constant

*apbRegPercent*  
%

*apbRegdB*  
dB

*apbRegAbs*  
Abs

**Description** This command selects the type of units the regulation algorithm uses to specify the Target Tolerance.**See Also** `ATS.Reg.TargetTolerance`, `ATS.Reg.TargetValue`**Example** See example for `ATS.Reg.SourceId`.**ATS.Reg.TargetValue****Property****Syntax** `ATS.Reg.TargetValue (ByVal Unit As String)`

**Data Type** Double Refer to the setting defined by the `ATS.Reg.TargetId` command (ID#) to determine the appropriate range of acceptable values.

Parameter	Name	Description
	<i>Unit</i>	Refer to the setting defined by the <code>ATS.Reg.TargetId</code> command (ID#) to determine the appropriate unit selections.

**Description** This command sets the Target value that the regulation algorithm attempts to obtain.**See Also** `ATS.Reg.TargetTolerance`, `ATS.Reg.TargetToleranceMode`**Example** See example for `ATS.Reg.SourceId`.

---

**ATS.Reg.TimeOut****Property**

<b>Syntax</b>	<code>ATS.Reg.Timeout</code>
<b>Data Type</b>	Double
<b>Description</b>	This command sets the period of time allowed to complete each regulation process.
<b>Example</b>	See example for <code>ATS.Reg.SourceId</code> .

User Notes

---

### ATS.Sweep.AbortTime

Property

**Syntax**            `ATS.Sweep.AbortTime`

**Data Type**        Double                    Time in seconds. Setting an abort time of zero seconds disables the abort function.

**Description**      This command defines the maximum time allowed for a sweep to complete after a sweep is started using any OLE command. *If the abort time is exceeded the current sweep is terminated.* If the `ATS.Data.ColSize` command returns a value less than the number of steps in the sweep then the sweep was aborted. This setting is not routinely monitored therefore accuracy may be in the seconds.

Note: This command is global and affects all subsequent sweeps. Care should be taken when using this command to disable it when finished.

**See Also**            `ATS.Sweep.Start`, `ATS.Sweep.StartWithAppend`,  
`ATS.Sweep.StartWithRepeat`, `ATS.Data.ColSize`

**Example**            `Sub Main`  
`Dim Steps As Integer`

```
Steps = 100
ATS.Application.NewTest
ATS2.AGen.OutputOn = True
ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
ATS2.AnalogIn.Source(apbChB) = apbAnalogInGenMon
ATS.Sweep.Source(1).Steps = Steps
ATS.Sweep.AbortTime = 5.0
ATS.Sweep.Start
```

```

If ATS.Data.ColSize(0, 0) < Steps Then
  With ATS.Prompt
    .Text = Chr$(13) & "Normal Sweep _
      Time exceeded" & Chr$(13) & _
      Chr$(13) & "Sweep Terminated"
    .FontSize = 8
    .Position(-1,-1,220,150)
    .Show
  Wait 3
  .Hide
  End With
End If
ATS.Sweep.AbortTime = 0.0
End Sub

```

---

## ATS.Sweep.Append

## Property

**Syntax**      **ATS.Sweep.Append**

**Data Type**    Boolean

*True*            Append data to current data in memory.  
*False*            Replace current data in memory.

**Description**    This command enables or disables appending data to the end of measurements contained in memory. If append is enabled the measurements in memory are retained and the next sweep will add additional measurements to memory. If append is disabled the measurements in memory are replaced by the next sweep data.

**See Also**        ATS.Sweep.Repeat, ATS.Sweep.StartWithAppend,  
 ATS.Sweep.StartWithRepeat

**Example**        Sub Main  
                   ATS.Application.NewTest  
                   ATS2.AGen.OutputOn = True  
                   ATS2.AGen.DACBandwidth = apbAGenBW\_60k  
                   **ATS.Sweep.CreateGraph** = True  
                   **ATS.Sweep.CreateTable** = False  
                   **ATS.Sweep.GraphType** = apbSweepGraphTypeData2onX  
                   ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon

```
ATS2.Inst.Analyzer.FuncFilterHP = apbAnlrHP22
ATS2.Inst.Analyzer.FuncFilterLP = apbAnlrLPFS2
```

'The commands in the following section could be replaced with commands for Data1-6

```
ATS.Sweep.Data(1).Id = 6343
ATS.Sweep.Data(1).Limits("None", 1, 1)
```

'The commands in the following section could be replaced with commands for Data2

```
ATS.Sweep.Data(1).LogLin = apbLin
ATS.Sweep.Data(1).AutoDiv = False
ATS.Sweep.Data(1).Div = 4
ATS.Sweep.Data(1).Autoscale = True
ATS.Sweep.Data(1).Top("V") = 1
ATS.Sweep.Data(1).Bottom("V") = 0
ATS.Sweep.Source(1).Start("Hz") = 20.0
ATS.Sweep.Source(1).Stop("Hz") = 60000.0
ATS.Sweep.PreSweepDelay = 0.2
ATS.Sweep.Start
ATS.Sweep.Append = True
ATS2.Inst.Analyzer.FuncFilterHP = apbAnlrHP100
ATS2.Inst.Analyzer.FuncFilterLP = apbAnlrLP20k
ATS.Sweep.Start
ATS2.Inst.Analyzer.FuncFilterHP = apbAnlrHP400
ATS2.Inst.Analyzer.FuncFilterLP = apbAnlrLP15K
ATS.Sweep.Start
ATS.Graph.OptimizeLeft
End Sub
```

---

## ATS.Sweep.CopySettings

## Method

**Syntax**            **ATS.Sweep.CopySettings** (ByVal *Data* As Constant)

**Data Type**        Constant

*apbData1ToData2*

Copy Data 1 settings to Data 2

*apbData2ToData1*

Copy Data 2 settings to Data 1

**Description** This command copies the Sweep panel settings from one Data to the other.

**Example**

```
Sub Main
  ATS.Application.NewTest
  ATS2.AGen.OutputOn = True
  ATS2.AnalogIn.Source (apbChA) = apbAnalogInGenMon
  ATS2.AnalogIn.Source (apbChB) = apbAnalogInGenMon
  ATS2.Inst.Selection = apbInstFFTAnalyzer
  ATS.Sweep.Data (1).Id = 6024
  ATS.Sweep.Data (2).Id = 6027
  ATS.Sweep.Source (1).Id = 5515
  ATS.Sweep.Start
  ATS.Graph.OptimizeLeft
  ATS.Graph.CopyToSweepPanel
  ATS.Sweep.CopySettings (apbData1ToData2)
  Wait 5
  ATS.Graph.OptimizeRight
  ATS.Graph.CopyToSweepPanel
  ATS.Sweep.CopySettings (apbData2ToData1)
End Sub
```

---

## ATS.Sweep.CreateGraph

## Property

**Syntax** `ATS.Sweep.CreateGraph`

**Data Type** Boolean

*True* Display a graph window when starting the sweep if a graph window is not displayed.

*False* Do not create a graph window when starting the sweep.

**Description** This command enables or disables creation of the graph window when a sweep is run.

**See Also** `ATS.Sweep.CreateTable`, `ATS.Sweep.GraphType`

**Example** See example for `ATS.Sweep.Append`.



---

**ATS.Sweep.CreateTable****Property**

<b>Syntax</b>	<b>ATS.Sweep.CreateTable</b>	
<b>Data Type</b>	Boolean	
	<i>True</i>	Display a Data Table window when starting the sweep if a Data Table window is not displayed.
	<i>False</i>	Do Not create a Data Table window when starting the sweep.
<b>Description</b>	This command enables or disables creation of the Data Table window when a sweep is run.	
<b>See Also</b>	ATS.Sweep.CreateGraph, ATS.Sweep.GraphType	
<b>Example</b>	See example for ATS.Sweep.Append.	

---

**ATS.Sweep.Data.AutoDiv****Property**

<b>Syntax</b>	<b>ATS.Sweep.Data (ByVal <i>Number</i> As Integer) .AutoDiv</b>	
<b>Data Type</b>	Boolean	
	<i>True</i>	Automatically select the number of divisions.
	<i>False</i>	Use the number of divisions defined by the ATS.Sweep.Data1.Div command.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Number</i>	Sweep Data selection number (1-2).
<b>Description</b>	This command enables or disables automatic selection of the number of linear vertical axis divisions displayed for Data 1 or Data 2.	
<b>See Also</b>	ATS.Sweep.Data(1).Div, ATS.Sweep.Data(1).LogLin	
<b>Example</b>	See example for ATS.Sweep.Append.	

## ATS.Sweep.Data.Autoscale

## Property

**Syntax** `ATS.Sweep.Data (ByVal Number As Integer) .Autoscale`

**Data Type** Boolean

*True* Autoscale graph vertical axis for Data (?).  
*False* Do Not Autoscale graph vertical axis for Data 1.

Parameter	Name	Description
	<i>Number</i>	Sweep Data selection number (1-2).

**Description** This command enables or disables automatic scaling of the graph vertical axis Top and Bottom values for Data 1 and Data 2. The Data 1 vertical axis is shown on the left axis of the graph and Data 2 on the right axis.

**Example** See example for `ATS.Sweep.Append`.

## ATS.Sweep.Data.Bottom

## Property

**Syntax** `ATS.Sweep.Data (ByVal Number As Integer) .Bottom (ByVal Unit As String)`

**Data Type** Double Enter a value that is to be displayed at the bottom of the graph left axis.

Parameter	Name	Description
	<i>Number</i>	Sweep Data selection number (1-2).
	<i>Unit</i>	Refer to the setting or reading defined by the <code>ATS.Sweep.Data (?) .Id</code> command to determine the appropriate unit selections.

**Description** This command defines the bottom value on the graph vertical axes located on the left and right sides of the graph window.

**See Also** `ATS.Sweep.Data.Top`

**Example** See example for `ATS.Sweep.Append`.

## ATS.Sweep.Data.Div

## Property

**Syntax** `ATS.Sweep.Data (ByVal Number As Integer) .Div`

**Data Type** Long                      Number of divisions displayed.

Parameter	Name	Description
	<i>Number</i>	Sweep Data selection number (1-2).

**Description** This command sets the number of divisions that are to be displayed for a linear vertical axis defined on Data 1 and Data 2. Automatic scaling of the vertical axes must be disabled using the `ATS.Sweep.Data.AutoDiv` comand.

**See Also** `ATS.Sweep.Data.AutoDiv`, `ATS.Sweep.Data.LogLin`

**Example** See example for `ATS.Sweep.Append`.

## ATS.Sweep.Data.Id

## Property

**Syntax** `ATS.Sweep.Data (ByVal Number As Constant) .Id`

**Data Type** Long                      Instrument Parameter ID#.

Parameter	Name	Description
	<i>Number</i>	Sweep Data selection number (1-6).

**Description** This command is used to select the instrument parameter, which will return readings for Data 1-6.

Refer to Appendix B to obtain instrument parameter identification numbers.

**Example** See example for `ATS.Sweep.Append`.

## ATS.Sweep.Data.Limit.Column

## Get Only Property

**Syntax** `ATS.Sweep.Data (ByVal DataNumber As Integer) .Limit (ByVal Limit As Constant) .Column`

**Data Type** Integer                      Column number.

Parameter	Name	Description
	<i>DataNumber</i>	Sweep Data selection number (1-6)
	<i>Limit</i>	apbUpper Upper limit. apbLower Lower limit.
<b>Description</b>	This command sets the Column number of the data used in the attached limit file for the specified Limit.	
<b>See Also</b>	ATS.Sweep.Data.Limits, ATS.Sweep.Data.Limit.FileName	

## ATS.Sweep.Data.Limit.FileName

### Get Only Property

<b>Syntax</b>	<b>ATS.Sweep.Data</b> (ByVal <i>DataNumber</i> As Integer) <b>.Limit</b> (ByVal <i>Limit</i> As Constant) <b>.FileName</b>	
<b>Data Type</b>	In++teger	Column number.
Parameter	Name	Description
	<i>DataNumber</i>	Sweep Data selection number (1-6)
	<i>Limit</i>	apbUpper Upper limit. apbLower Lower limit.
<b>Description</b>	This command returns the File Name used for the Upper and or Lower Limit of a Sweep Data.	
<b>See Also</b>	ATS.Sweep.Data.Limits, ATS.Sweep.Data.Limit.Column	

## ATS.Sweep.Data.Limits

### Method

<b>Syntax</b>	<b>ATS.Sweep.Data</b> (ByVal <i>Number</i> As Integer) <b>.Limits</b> (ByVal <i>PathName</i> As String, ByVal <i>Column</i> As Integer, ByVal <i>Upper</i> As Boolean)	
Parameter	Name	Description
	<i>Number</i>	Sweep Data selection number (1-6).
	<i>PathName</i>	Any valid DOS path and file name. The file must be an ATS limit file (.atsl). Enter "None" for the file name to remove the limit file from Data (?).

	<i>Column</i>	1 = Data 1 measurements. 2 = Data 2 measurements. 3 = Data 3 measurements. 4 = Data 4 measurements. 5 = Data 5 measurements. 6 = Data 6 measurements.
	<i>Upper</i>	True = Upper Limit. False = Lower Limit.
<b>Result</b>	Boolean	
	<i>True</i>	File attachment successful.
	<i>False</i>	File attachment failed.
<b>Description</b>	This command attaches or removes a limit file from Data 1-6 for upper or lower limit comparisons.	
<b>See Also</b>	ATS.Sweep.Recompare	
<b>Example</b>	<pre> Sub Main     ATS.LogFile.Enable = False     ATS.File.OpenTest "CODEC.ATS2"     ATS.Sweep.Start     ATS.File.SaveDataAs "MASK.ATSL"     ATS.Application.NewData     ATS2.Inst.FastTest.Mode = apbFastTestSpectrum <b>ATS.Sweep.Reprocess</b>     ATS.Sweep.Source1.Table ("CODEC.ATSS", 0)     ATS2.Inst.FastTest.Mode = apbFastTestResponse <b>ATS.Sweep.Reprocess</b> <b>ATS.Sweep.Data(1).Limits</b> ("MASK.ATSL", 1, True) <b>ATS.Sweep.Data(3).Limits</b> ("Mask.ATSL", 1, True)     ATS2.Inst.FastTest.Mode = apbFastTestDistortion <b>ATS.Sweep.Reprocess</b>     ATS2.Inst.FastTest.Mode = apbFastTestNoise <b>ATS.Sweep.Reprocess</b> End Sub </pre>	

---

## ATS.Sweep.Data.LogLin

## Property

**Syntax**      **ATS.Sweep.Data** (ByVal *Number* As Integer) .**LogLin**

<b>Data Type</b>	Constant <i>apbLog</i>  <i>apbLin</i>	Logarithmic vertical axis.  Linear vertical axis.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Number</i>	Sweep Data selection number (1-2).
<b>Description</b>	This command determines the Data 1 and Data 2 vertical axis data scaling type.	
<b>See Also</b>	ATS.Sweep.Data.Div, ATS.Sweep.Data.AutoDiv	
<b>Example</b>	See example for ATS.Sweep.Append.	

## ATS.Sweep.Data.Top

## Property

<b>Syntax</b>	<b>ATS.Sweep.Data</b> (ByVal <i>Number</i> As Integer) . <b>Top</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	Enter a value that is to be displayed at the top of the graph left axis.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Number</i>	Sweep Data selection number (1-2).
	<i>Unit</i>	Refer to the setting or reading defined by the ATS.Sweep.Data.Id command to determine the appropriate unit selections.
<b>Description</b>	This command defines the top value on the graph vertical axes located on the left and right sides of the graph window.	
<b>See Also</b>	ATS.Sweep.Data.Bottom	
<b>Example</b>	See example for ATS.Sweep.Append.	

## ATS.Sweep.External.StartOnRule

## Property

**Syntax** `ATS.Sweep.External.StartOnRule`

**Data Type** Integer

- 0 **Within Start Tolerance:** (Default) Data collection begins as soon as a settled reading is detected that is within the Start value plus or minus the Spacing tolerance as specified by the `ATS.Sweep.Source.Start`, and `ATS.Sweep.Source.Spacing` commands.
- 1 **Beyond Start Value:** Data collection begins as soon as a settled reading is detected that is within the Start and Stop values as specified by the `ATS.Sweep.Source.Start` and `ATS.Sweep.Source.Stop` commands. Additional readings are retained as long as the readings satisfy the sweep direction and Spacing requirement as specified by the `ATS.Sweep.Source.Spacing` command.
- 2 **Any Settled Reading:** Data collection begins as soon as a settled reading is detected. Additional readings are retained as long as the readings satisfy the sweep direction and Spacing requirements as specified by the `ATS.Sweep.Source.Start`, `ATS.Sweep.Source.Stop`, and `ATS.Sweep.Source.Spacing` commands.

**Description** This command selects the external sweep Start On rule.

**See Also** `ATS.Sweep.Source.Start`, `ATS.Sweep.Source.Stop`, `ATS.Sweep.Source.Spacing`

**Example**

```
Sub Main
    AP.Application.VisibleMacroEditor False
    Begin Dialog UserDialog 390,231,"External Sweep
Rules" ' %GRID:10,7,1,1
        Text 20,7,350,70,"Sets External Sweep
""Start-On"" rules. Default setting starts sweep
when the Sourcel meter reading is within the Start
"&Chr(177)&" Spacing field settings. This setting is
not saved with the test. It is global and applies to
```

```

ALL external tests until changed or ATS is restarted.
Current ""Start-On"" setting is indicated."
  GroupBox 20,84,350,98,"  START - ON  RULE  "
  Text 30,98,320,14, _
    "Starts Sweep when Source1 meter reading _
    is - ",.Text1
  OptionGroup .StartOnRule
    OptionButton 50,119,300,14, _
      "Within Start " & Chr(177) & _
      " Spacing  settings."
    OptionButton 50,140,300,14, _
      "Beyond the Start setting."
    OptionButton 50,161,300,14, _
      "Any settled reading."
  OKButton 20,196,140,21
  CancelButton 280,196,90,21
End Dialog
Dim dlgXstartOn As UserDialog
dlgXstartOn.StartOnRule =
AP.Sweep.External.StartOnRule
'Get current setting and set dialog option buttons to
match
  If Dialog(dlgXstartOn) = 0 Then Exit Sub
    'If cancel, end
  Select Case(dlgXstartOn.StartOnRule)
    Case 0      '+-Tolerance
      AP.Sweep.External.StartOnRule = _
        apbExtWithinTolerance
    Case 1      'Beyond Start setting
      AP.Sweep.External.StartOnRule = - _
        apbExtBeyondStart
    Case 2      'No requirement
      AP.Sweep.External.StartOnRule = _
        apbExtNone
  End Select
End Sub

```



---

**ATS.Sweep.GraphType****Property****Syntax**      **ATS . Sweep . GraphType****Data Type**      Constant*apbSweepGraphTypeXY*

X - Y mode. Data 1-6 measurements are displayed on the vertical axis and Source settings are displayed on the horizontal axis.

*apbSweepGraphTypeData2onX*

X - Y Data2 on X mode. Data 1, and 3-6 measurements are displayed on the vertical axis and Data 2 readings are displayed on the horizontal axis.

**Description**      This command selects the graph display mode. The `ATS . Sweep . Data . Id` must be defined.**See Also**      `ATS . Sweep . Data . Id`**Example**      See example for `ATS . Sweep . Append`.

---

**ATS.Sweep.IsRunning****Property****Syntax**      **ATS . Sweep . IsRunning****Result**      Boolean*True*

Sweep process running.

*False*

Sweep process not running.

**Description**      This command returns the status of the Sweep process.**Example**      See example for `ATS . Sweep . Stop`.

---

**ATS.Sweep.PreSweepDelay****Property****Syntax**      **ATS . Sweep . PreSweepDelay****Data Type**      Double      0.0 to 3.0 sec.

**Description** This command sets a user-controllable time delay value inserted after the `ATS.Sweep.Start` command is executed, before the first data point is taken. This can be valuable when the device under test needs a certain amount of setup time before it operates normally, or to allow for full autoranging and other time within the instrument. In nested sweeps, this Pre-Sweep Delay is inserted before the start of each sweep of the test.

The Pre-Sweep Delay field is located on the right half of the large version of the Sweep panel, below the Data 3-Data 6 Limits buttons.

**See Also** `ATS.Sweep.Start`

**Example** See example for `ATS.Sweep.Append`.

---

## ATS.Sweep.Recompare

### Method

**Syntax** `ATS.Sweep.Recompare`

**Result** Boolean

*True*                      Recompare successful.  
*False*                     Recompare failed.

**Description** This command causes any sweep result currently in memory to be regraphed and compared to limits if limit files are attached to any Data (Data 1 - Data 6) variable via the test configuration or usage of the `ATS.Sweep.Data.Limits` command.

This command is equivalent to F7 in ATS.

**See Also** `ATS.Sweep.Data.Limits`

---

## ATS.Sweep.Repeat

### Property

**Syntax** `ATS.Sweep.Repeat`

**Data Type** Boolean

*True*                        Repeat sweep continuously.  
*False*                      Do not repeat sweep continuously.

**Description** This command enables or disables repeating the currently defined sweep indefinitely.

**See Also** `ATS.Sweep.Append`, `ATS.Sweep.StartWithAppend`,  
`ATS.Sweep.StartWithRepeat`

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS.Application.PanelOpen(apbSweep, apbSmall)
    ATS.Prompt.Text = "Press Continue to Stop Sweep."
    ATS.Prompt.FontSize = 8
    ATS.Prompt.Position(-1,-1,190,120)
    Begin Dialog UserDialog 310,154,"Sweep Controler"
        PushButton 100,7,100,21,"Single _
            sweep",.PushButton1
        PushButton 30,35,250,21,"Single sweep and _
            Append",.PushButton3
        PushButton 30,56,250,21,"Start repeating _
            sweep",.PushButton2
        PushButton 30,77,250,21,"Start repeating sweep _
            with Append",.PushButton4
        CancelButton 60,119,190,21
    End Dialog
    Dim dlg As UserDialog

    DisplayDialog:
    Select Case Dialog (dlg)
        Case 0
            ATS.Sweep.Append = False
            ATS.Sweep.Repeat = False
        End
        Case 1
            ATS.Sweep.Append = False
            ATS.Sweep.Repeat = False
            Sweep.Start
        Case 2
            ATS.Sweep.Repeat = False
            ATS.Sweep.StartWithAppend
        Case 3
```

```

        ATS.Sweep.Append = False
        ATS.Prompt.ShowWithContinueAndStopSweep
        ATS.Sweep.StartWithRepeat
    'Start sweep
        ATS.Sweep.Repeat = False
Case 4
    ATS.Sweep.Append = True
    ATS.Sweep.Repeat = True
    ATS.Prompt.ShowWithContinueAndStopSweep
    ATS.Sweep.Start
    'Start sweep
    ATS.Sweep.Repeat = False
End Select
GoTo DisplayDialog
End Sub

```

---

## ATS.Sweep.Reprocess

## Method

**Syntax**      **ATS . Sweep . Reprocess**

**Result**      Boolean

*True*                  Reprocess successful.  
*False*                 Reprocess failed.

**Description**      This command instructs ATS to cause the third phase of the following process to be performed.

FFT-based (batch mode) DSP programs have three distinct, sequential phases to their operation.

First, data is accumulated into the acquisition buffer until the buffer is filled to the specified acquisition length.

Second, a Fast Fourier Transform (FFT) is performed to obtain amplitude (and sometimes phase) versus frequency data which is stored in a different memory buffer from the acquired signal (amplitude versus time).

Third, a post-processed version of the amplitude versus time or amplitude versus frequency data (depending upon sweep Source 1

and Data 1 or 2) is transmitted from the DSP module in the test system to the computer for graphing by ATS software.

This command is equivalent to Ctrl+F6 in ATS.

**Example** See example for `ATS.Sweep.Data.Limits`.

---

## ATS.Sweep.Retransform

## Method

**Syntax** `ATS.Sweep.Retransform`

**Result** Boolean

*True*                   Retransform successful.  
*False*                   Retransform failed.

**Description** This command instructs ATS to cause the second and third phases of the following process to be performed.

FFT-based (batch mode) DSP programs have three distinct, sequential phases to their operation.

First, data is accumulated into the acquisition buffer until the buffer is filled to the specified acquisition length.

Second, a Fast Fourier Transform (FFT) is performed to obtain amplitude (and sometimes phase) versus frequency data which is stored in a different memory buffer from the acquired signal (amplitude versus time).

Third, a post-processed version of the amplitude versus time or amplitude versus frequency data (depending upon sweep Source 1 and Data 1 or 2) is transmitted from the DSP module in the test system to the computer for graphing by ATS software.

This command is equivalent to F6 in ATS.

**Example**

```
Sub Main
  ATS.Application.NewTest
  ATS2.AGen.OutputOn = True
  ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
  ATS2.Inst.Selection = apbInstFFTAnalyzer
  ATS2.Inst.FFT.TransformLength = apbFFT16k
```

```

ATS2.Inst.FFT.Window = apbFFTBlackmanHarris
ATS.Sweep.Data(1).Id = 6024
ATS.Sweep.Source1.Id = 5515
ATS.Sweep.Start
ATS.Sweep.Append = True
ATS2.Inst.FFT.Window = apbFFTHann
ATS.Sweep.Retransform
ATS2.Inst.FFT.Window = apbFFTFlatTop
ATS.Sweep.Retransform
ATS2.Inst.FFT.Window = apbFFTEquiripple
ATS.Sweep.Retransform
ATS2.Inst.FFT.Window = apbFFTNone
ATS.Sweep.Retransform
ATS.Graph.OptimizeLeft
End Sub

```

---

## ATS.Sweep.SinglePoint

## Property

**Syntax**      **ATS.Sweep.SinglePoint**

**Data Type**    Boolean

*True*            Enable single point sweep.  
*False*            Disable single point sweep.

**Description**    This command sets the Source 1 Sweep to Single Point mode. When a sweep is initiated (`ATS.Sweep.Start`) the Data Editor will be automatically displayed and a single measurement taken at the Sweep Start value of Source 1.

**See Also**        `ATS.Sweep.Source1.Start`

**Example**        Sub Main  
                   `ATS.Application.NewTest`  
                   `ATS2.AGen.OutputOn = True`  
                   `ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon`  
                   `ATS2.Inst.Selection = apbInstFFTAnalyzer`  
                   `ATS.Sweep.Repeat = False`  
                   **`ATS.Sweep.SinglePoint = False`**  
                   **`ATS.Sweep.Stereo = False`**  
                   **`ATS.Sweep.Timeout("sec") = 3`**

```

ATS.Sweep.Source(1).Id = 5015
ATS.Sweep.Source(1).LogLin = 1
ATS.Sweep.Source(1).Start("Hz") = 20000
ATS.Sweep.Source(1).Stop("Hz") = 20
ATS.Sweep.Source(1).Steps = 15
ATS.Sweep.Source(1).AutoDiv = False
ATS.Sweep.Source(1).Div = 10
ATS.Sweep.Source(2).Id = 5052
ATS.Sweep.Source(2).LogLin = apbLin
ATS.Sweep.Source(2).Start("Vrms") = 5
ATS.Sweep.Source(2).Steps = 2
ATS.Sweep.Source(2).Stop("Vrms") = 1
ATS.Sweep.Start
End Sub
    
```

---

## ATS.Sweep.Source.AutoDiv

## Property

**Syntax** `ATS.Sweep.Source (ByVal Number As Integer).AutoDiv`

**Data Type** Boolean

*True* automatically select the number of divisions.  
*False* Use the number of divisions defined by the  
 ATS.Sweep.Source1.Div command.

Parameter	Name	Description
	<i>Number</i>	Sweep Source selection number (1 only).

**Description** This command enables or disables automatic selection of the number of linear horizontal axis divisions displayed for Source 1 sweeps.

**See Also** `ATS.Sweep.Source.Div`, `ATS.Sweep.Source.LogLin`

**Example** See example for `ATS.Sweep.SinglePoint`.

---

## ATS.Sweep.Source.Div

## Property

**Syntax** `ATS.Sweep.Source (ByVal Number As Integer).Div`

<b>Data Type</b>	Long	Number of divisions displayed.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Number</i>	Sweep Source selection number (1 only).
<b>Description</b>	This command sets the number of divisions that are to be displayed for a linear horizontal axis for a Source 1. The <code>ATS.Sweep.Source1.AutoDiv</code> must be disabled.	
<b>See Also</b>	<code>ATS.Sweep.Source.AutoDiv</code> , <code>ATS.Sweep.Source.LogLin</code>	
<b>Example</b>	See example for <code>ATS.Sweep.SinglePoint</code> .	

## ATS.Sweep.Source.EndOn

## Property

<b>Syntax</b>	<code>ATS.Sweep.Source</code> (ByVal <i>Number</i> As Integer) . <code>EndOn</code> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	Refer to the setting or reading defined by the <code>ATS.Sweep.Source1.Id</code> command to determine the appropriate range of acceptable values.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Number</i>	Sweep Source selection number (1 only).
	<i>Unit</i>	Refer to the setting or reading defined by the <code>ATS.Sweep.Source.Id</code> command to determine the appropriate unit selections.
<b>Description</b>	<p>This command sets the Sweep End value for an external sweep. The sweep will be considered to have finished when the Source 1 parameter reverses its direction (starts to change in the direction from Stop to Start) to the End On value.</p> <p>It is frequently necessary to make and graph a series of measurements where some external, uncontrollable source is the independent variable. Common examples include frequency response measurements or other swept tests where the sweeping signal is pre-recorded on a test tape or test CD, or testing of a transmission link where a remote generator (not under control of ATS software) is providing the signal. In these cases, ATS software cannot</p>	



control the values, direction of progression (high to low versus low to high), or dwell times of the signal. ATS can, however, measure the changing parameter of the incoming signal (usually frequency but sometimes level) and use those measurements as the X-axis calibration. This mode of operation, where a measurement (Reading) drives the data-taking process and calibrates the X-axis, is called External Sweep.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS.Sweep.Source(1).Id = 6009
    ATS.Sweep.Source(1).EndOn("Hz") = 2500
    ATS.Sweep.Source(1).MinLevelID = 6341
    ATS.Sweep.Source(1).MinLevel("dBu") = -40
    ATS.Sweep.Source(1).Spacing("%") = 3
    ATS.Sweep.Start
End Sub
```

**ATS.Sweep.Source.Id****Property**

**Syntax**      **ATS.Sweep.Source** (ByVal *Number* As Constant) .**Id**

**Data Type**      Long                      Instrument Parameter ID#.

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Number</i>	Sweep Source selection number (1-2).

**Description**      This command is used to select the instrument parameter which will define settings or return readings, in the case of external sweeps, for Source 1 and Source 2.

Refer to Appendix B to obtain instrument parameter identification numbers.

**Example**              See example for `ATS.Sweep.SinglePoint`.

**ATS.Sweep.Source.LogLin****Property**

**Syntax**              **ATS.Sweep.Source** (ByVal *Number* As Integer) .**LogLin**

<b>Data Type</b>	Integer				
	<p>0                      Logarithmic horizontal axis and step type.</p> <p>1                      Linear horizontal axis and step type.</p>				
<b>Parameter</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>Number</i></td> <td>Sweep Source selection number (1 only).</td> </tr> </tbody> </table>	Name	Description	<i>Number</i>	Sweep Source selection number (1 only).
Name	Description				
<i>Number</i>	Sweep Source selection number (1 only).				
<b>Description</b>	This command determines the Source 1 horizontal axis type and the sweep step type.				
<b>See Also</b>	ATS.Sweep.Source.Div, ATS.Sweep.Source.AutoDiv				
<b>Example</b>	See example for ATS.Sweep.SinglePoint.				

## ATS.Sweep.Source.MinLevel

## Property

<b>Syntax</b>	<b>ATS.Sweep.Source</b> (ByVal <i>Number</i> As Integer) <b>.MinLevel</b> (ByVal <i>Unit</i> As String)							
<b>Data Type</b>	Double	Refer to the reading defined by the <code>ATS.Sweep.Source.MinLevelId</code> command to determine the appropriate range of acceptable values.						
<b>Parameter</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>Number</i></td> <td>Sweep Source selection number (1 only).</td> </tr> <tr> <td><i>Unit</i></td> <td>Refer to the reading defined by the <code>ATS.Sweep.Source.MinLevelId</code> command to determine the appropriate unit selections.</td> </tr> </tbody> </table>	Name	Description	<i>Number</i>	Sweep Source selection number (1 only).	<i>Unit</i>	Refer to the reading defined by the <code>ATS.Sweep.Source.MinLevelId</code> command to determine the appropriate unit selections.	
Name	Description							
<i>Number</i>	Sweep Source selection number (1 only).							
<i>Unit</i>	Refer to the reading defined by the <code>ATS.Sweep.Source.MinLevelId</code> command to determine the appropriate unit selections.							
<b>Description</b>	<p>This command sets the minimum input signal level at which measurements will be taken during an external sweep (reading instead of setting at Source 1). The purpose of this command is to avoid taking measurements during the "dead time" between tracks of a test tape or test CD, when noise still produces some finite signal level.</p> <p>It is frequently necessary to make and graph a series of measurements where some external, uncontrollable source is the independent variable. Common examples include frequency response measurements or other swept tests where the sweeping signal is pre-recorded on a test tape or test CD, or testing of a</p>							

transmission link where a remote generator (not under control of ATS software) is providing the signal. In these cases, ATS software cannot control the values, direction of progression (high to low versus low to high), or dwell times of the signal. ATS can, however, measure the changing parameter of the incoming signal (usually frequency but sometimes level) and use those measurements as the X-axis calibration. This mode of operation, where a measurement (Reading) drives the data-taking process and calibrates the X-axis, is called External Sweep.

**See Also** `ATS.Sweep.Data.Id`, `ATS.Sweep.MinLevelSource`

---

## ATS.Sweep.Source.MinLevelId

## Property

**Syntax** `ATS.Sweep.Source` (ByVal *Number* As Integer) **.MinLevelId**

**Data Type** Long Instrument Parameter ID#.

Parameter	Name	Description
	<i>Number</i>	Sweep Source selection number (1 only).

**Description** This command is used to select the instrument parameter which will define settings or return readings, in the case of external sweeps, for Source 1.

Refer to Appendix B to obtain instrument parameter identification numbers.

---

## ATS.Sweep.Source.Multiply

## Property

**Syntax** `ATS.Sweep.Source` (ByVal *Number* As Integer) **.Multiply**

**Data Type** Double

Parameter	Name	Description
	<i>Number</i>	Sweep Source selection number (1-2).

**Description** This command sets the Source(?) Log Sweep Multiply factor used to determine the next Source(?) sweep setting.

**See Also** `ATS.Sweep.Source.Start`, `ATS.Sweep.Source.Stop`, `ATS.Sweep.Source.LogLin`, `ATS.Sweep.Source.Steps`

---

## ATS.Sweep.Source.Spacing

## Property

**Syntax** `ATS.Sweep.Source` (ByVal *Number* As Integer) `.Spacing` (ByVal *Unit* As String)

**Data Type** Double

Parameter	Name	Description
	<i>Number</i>	Sweep Source selection number (1 only).
	<i>Unit</i>	% unit only.

**Description** This command sets the minimum change of the Source 1 Reading (ID#) Property parameter required to allow an additional external sweep measurement to be taken. This setting is only available for external sweeps.

**Example** See example for `ATS.Sweep.Source, EndOn`.

---

## ATS.Sweep.Source.Start

## Property

**Syntax** `ATS.Sweep.Source` (ByVal *Number* As Integer) `.Start` (ByVal *Unit* As String)

**Data Type** Double Refer to the setting or reading defined by the `ATS.Sweep.Source.Id` command to determine the appropriate range of acceptable values.

Parameter	Name	Description
	<i>Number</i>	Sweep Source selection number (1-2).
	<i>Unit</i>	Refer to the setting or reading defined by the <code>ATS.Sweep.Source.Id</code> command to determine the appropriate unit selections.

- Description** This command sets the first setting value to be sent to the instrument parameter specified as Source(?).
- See Also** `ATS.Sweep.Source.Stop`
- Example** See example for `ATS.Sweep.Append`.

## ATS.Sweep.Source.Steps

## Property

- Syntax** `ATS.Sweep.Source (ByVal Number As Integer) .Steps`
- Data Type** Long
- Parameter**

Name	Description
<i>Number</i>	Sweep Source selection number (1-2).
- Description** This command sets the number of Source(?) steps that a log or linear sweep makes between the Source Start and Stop values.
- See Also** `ATS.Sweep.Source.Start`, `ATS.Sweep.Source.Stop`, `ATS.Sweep.Source.LogLin`, `ATS.Sweep.Source.StepSize`
- Example** See example for `ATS.Sweep.SinglePoint`.

## ATS.Sweep.Source.StepSize

## Property

- Syntax** `ATS.Sweep.Source (ByVal Number As Integer) .StepSize (ByVal Unit As String)`
- Data Type** Double          Source step size.
- Description** This command sets the Source Linear Sweep Step Size used to determine the next Source sweep setting.
- Parameter**

Name	Description
<i>Number</i>	Sweep Source selection number (1-2).
<i>Unit</i>	Refer to the setting or reading defined by the <code>ATS.Sweep.Source.Id</code> command to determine the appropriate unit selections.

**See Also** `ATS.Sweep.Source.Start`, `ATS.Sweep.Source.Stop`,  
`ATS.Sweep.Source.LogLin`, `ATS.Sweep.Source.Steps`

## ATS.Sweep.Source.Stop

**Property**

**Syntax** `ATS.Sweep.Source` (ByVal *Number* As Integer) **.Stop** (ByVal *Unit* As String)

**Data Type** Double Refer to the setting or reading defined by the `ATS.Sweep.Source.Id` command to determine the appropriate range of acceptable values.

Parameter	Name	Description
	<i>Number</i>	Sweep Source selection number (1-2).
	<i>Unit</i>	Refer to the setting or reading defined by the <code>ATS.Sweep.Source.Id</code> command to determine the appropriate unit selections.

**Description** This command sets the last setting value to be sent to the instrument parameter specified as `Source(?)`.

**See Also** `ATS.Sweep.Source.Start`

**Example** See example for `ATS.Sweep.Append`.

## ATS.Sweep.Source.Table

**Method**

**Syntax** `ATS.Sweep.Source` (ByVal *Number* As Integer) **.Table** (ByVal *FileName* As String, ByVal *Column* As Integer)

Parameter	Name	Description
	<i>Number</i>	Sweep Source selection number (1 only).
	<i>FileName</i>	Any valid DOS path and file name. The file must be an ATS sweep file (.atss). Enter "None" for the file name to remove the sweep file from <code>Source1</code> .
	<i>Column</i>	0 = Source 1 settings. 1 = Data 1 measurements. 2 = Data 2 measurements.

- 3 = Data 3 measurements.
- 4 = Data 4 measurements.
- 5 = Data 5 measurements.
- 6 = Data 6 measurements.
- 7 = Source 2 settings.

**Result** Boolean

*True* File attachment successful.

*False* File attachment failed.

**Description** This command attaches a sweep file to Source 1. Values in the file will be used as Source 1 settings, rather than Start, Stop, Steps, and Multiply, or StepSize values. The Start and Stop values will continue to be used to define the horizontal end points of the graph.

## ATS.Sweep.Source.TableColumn

### Get Only Property

**Syntax** `ATS.Sweep.Source (ByVal DataNumber As Integer) .TableColumn`

**Data Type** Integer Column number.

Parameter	Name	Description
	<i>DataNumber</i>	Sweep Data selection number (1-6)

**Description** This command sets the column number of the data in the attached Sweep Table file (.atss) to be used as the Sweep Table.

**See Also** `ATS.Sweep.Source.TableFileName`

## ATS.Sweep.Source.TableFileName

### Get Only Property

**Syntax** `ATS.Sweep.Source (ByVal DataNumber As Integer) .TableFileName`

**Data Type** Integer Column number.

Parameter	Name	Description
	<i>DataNumber</i>	Sweep Data selection number (1-6)

**Description** This command sets the File Name used for the Sweep panel Sweep Table.

**See Also** `ATS.Sweep.Source.TableColumn`

---

## ATS.Sweep.Spectrum

## Method

**Syntax** `ATS.Sweep.Spectrum`

**Result** Boolean

*True* Change to Spectrum Display successful.  
*False* Display change not successful.

**Description** This command configures the Sweep Panel to produce a Spectrum display when a sweep is run.

### **New Test configuration functionality:**

In this situation the user has not defined the Sweep Panel to display a Spectrum but has selected from one of the Digital Analyzer selections listed below. If the user has not selected from one of the Digital Analyzer selections listed below this command is not active.

When this command is executed default values are automatically entered into the sweep panel settings to setup the sweep to display the default Spectrum when run. Each Digital Analyzer selection listed below has it's own default sweep panel settings for a frequency domain display.

### **User defined test functionality:**

In this situation the user has loaded a previously saved test. If the user has redefined any of the default sweep panel settings for any or all of the Digital Analyzer selections and then saved the settings as a test then all of the settings for all of the Digital Analyzer selections will be restored when the test is loaded. The user can then switch between any of the Digital Analyzer selections listed below and the previously defined settings will be restored.

### **Digital Analyzer selections:**



FFT spectrum analyzer  
 INTERVU Digital interface analyzer  
 Multitone audio analyzer

**See Also** `ATS.Sweep.Waveform`

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.AnalogIn.Source(apbChB) = apbAnalogInGenMon
    ATS2.Inst.Selection = apbInstFFTAnalyzer
    ATS.Sweep.Spectrum
    ATS.Application.Page = 2
    ATS.Sweep.Start
    Wait 5
    ATS.Sweep.Waveform
End Sub
```

---

## ATS.Sweep.Start

**Method**

**Syntax** `ATS.Sweep.Start`

**Result** Boolean

*True* Sweep completed successfully.  
*False* Sweep terminated abnormally.

**Description** This command initiates a sweep.

**Note:** When using this command from an external application execution of additional commands will not be held off if the `ATS.Sweep.Repeat` command is set to `True`. The `ATS.Sweep.Repeat` command is also affected by the `ATS.Sweep.StartWithRepeat` command.

**See Also** `ATS.Sweep.StartWithAppend`,  
`ATS.Sweep.StartWithRepeat`

**Example** Sub Main

```

ATS.File.OpenTest ("FRQ-RESP.ATS2")
ATS.Sweep.Start
ATS.File.SaveDataAs ("FRQ-RESP.ATSA")
ATS.File.OpenTest ("THD-FRQ.ATS2")
ATS.Sweep.Start
ATS.File.SaveDataAs ("THD-FRQ.ATSA")
ATS.File.OpenTest ("RESIDNOI.ATS2")
ATS.Sweep.Start
ATS.File.SaveDataAs ("RESIDNOI.ATSA")
End Sub

```

---

## ATS.Sweep.StartNoWait

## Method

**Syntax**            **ATS.Sweep.StartNoWait**

**Result**            Boolean

*True*                Sweep process started successfully.

*False*               Sweep process not started successfully.

**Description**     This command initiates a sweep process and then continues macro execution.

**Example**           See example for `ATS.Sweep.Stop`.

---

## ATS.Sweep.StartWithAppend

## Method

**Syntax**            **ATS.Sweep.StartWithAppend**

**Description**     This command initiates a sweep in append mode which is equivalent to pressing the Ctrl+F9 function key.

**See Also**           `ATS.Sweep.Start`, `ATS.Sweep.StartWithRepeat`

**Example**           See example for `ATS.Sweep.SinglePoint`.

---

**ATS.Sweep.StartWithRepeat****Method**

<b>Syntax</b>	<code>ATS.Sweep.StartWithRepeat</code>
<b>Description</b>	This command initiates a sweep in repeat mode which is equivalent to pressing the Alt+F9 function key.
<b>See Also</b>	<code>ATS.Sweep.Start</code> , <code>ATS.Sweep.StartWithAppend</code>
<b>Example</b>	See example for <code>ATS.Sweep.Repeat</code> .

---

**ATS.Sweep.Stereo****Property**

<b>Syntax</b>	<code>ATS.Sweep.Stereo</code>
<b>Data Type</b>	Boolean
	<i>True</i> Enable Stereo Sweep
	<i>False</i> Disable Stereo Sweep.
<b>Description</b>	This command enables or disables the stereo sweep feature on the Sweep panel.
<b>Example</b>	See example for <code>ATS.Sweep.SinglePoint</code> .

---

**ATS.Sweep.Stop****Method**

<b>Syntax</b>	<code>ATS.Sweep.Stop</code>
<b>Result</b>	Boolean
	<i>True</i> Sweep terminated successfully.
	<i>False</i> Sweep not terminated.
<b>Description</b>	This command terminates a running sweep.
<b>See Also</b>	<code>ATS.Sweep.IsRunning</code>
<b>Example</b>	<pre>Dim Halt As Boolean Sub Main     Halt = False</pre>

```

ATS.Application.NewTest
ATS2.AGen.OutputOn = True
ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
ATS.Sweep.Source1.Steps = 200
ATS.Application.SetWatchDogTimer(1, 5.0, False)
ATS.Sweep.StartNoWait
Do
    'nothing
    Loop While Halt = False
End Sub
Public Sub ATSEvent_OnWatchDogTimeout _
    (ByVal Id As Long)
    If Id = 1 Then
        Halt = True
        If ATS.Sweep.IsRunning = True Then
            ATS.Sweep.Stop
            Debug.Print "Sweep Stopped"
        End If
    End If
End Sub

```

## ATS.Sweep.Timeout

## Property

**Syntax**      **ATS.Sweep.Timeout** (ByVal *Unit* As String)

**Data Type**      Double      Timeout values of 0 to 3000 seconds (50 minutes) are allowed.

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
		<i>Unit</i>

**Description**      This command sets the timeout used during settling comparisons. If settling cannot be achieved during the Timeout duration, the average of its last 6 readings is computed and returned. Timeout serves as a "safety valve" to avoid excessive delays or hang-up when the data has more variation that present settling parameters will accept.

In a graph display, each timeout point is indicated by a white T at the upper margin of the graph, directly above the plotted point. In the Data Editor, each timeout point is indicated by the letter T following

the data. In the Log File, the Pass/Fail message (if enabled) shows the total number of timeouts which occurred during a sweep. However, a timeout is not treated as a failure if the eventual averaged data was within limits. The Log File may also include a line for each measured point which timed out during the sweep resulting in a row showing the measured value and a letter T.

See Appendix A for Settling Algorithm and parameter name descriptions.

**Example** See example for `ATS.Sweep.SinglePoint`.

---

## ATS.Sweep.Waveform

## Method

**Syntax** `ATS.Sweep.Waveform`

**Result** Boolean

*True* Change to Waveform Display successful.  
*False* Display change not successful.

**Description** This command configures the Sweep Panel to produce a Waveform display when a sweep is run.

### **New Test configuration functionality:**

In this situation the user has not defined the Sweep Panel to display a Waveform but has selected from one of the Digital Analyzer selections listed below. If the user has not selected from one of the Digital Analyzer selections listed below this command is not active.

When this command is executed default values are automatically entered into the sweep panel settings to setup the sweep to display the default Waveform when run. Each Digital Analyzer selection listed below has its own default sweep panel settings for a time domain display.

### **User defined test functionality:**

In this situation the user has loaded a previously saved test. If the user has redefined any of the default sweep panel settings for any or all of the Digital Analyzer selections and then saved the settings as a

test then all of the settings for all of the Digital Analyzer selections will be restored when the test is loaded. The user can then switch between any of the Digital Analyzer selections listed below and the previously defined settings will be restored.

**Digital Analyzer selections:**

FFT spectrum analyzer

INTERVU Digital interface analyzer

FASTTEST Multitone audio analyzer

**See Also**      `ATS.Sweep.Spectrum`

**Example**      See example for `ATS.Sweep.Spectrum`.

# Chapter 18

## Analog Generator

### ATS2.AGen.Ampl

### Property

**Syntax**      **ATS2.AGen.Ampl** (ByVal *Channel* As Constant, ByVal *Unit* As String)

**Data Type**      Double

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Vrms, Vp, Vpp, dBu, dBV, dBr, W, dBrlnv

**Description**      This command sets the Analog Generator amplitude.

**Example**

```
Sub Main
    Dim reading(30)
    ndx = 0
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.AGen.Ampl(apbChA, "Vrms") = 5
    For NewFreq = 20 To 20e3 Step (20e3 - 20)/30
        ATS2.AGen.Freq(apbChA, "Hz") = NewFreq
        reading(ndx) = ATS2.Inst.Analyzer.FuncRdg _
            (apbChA, "V")
        ndx = ndx + 1
    Next
End Sub
```

**ATS2.AGen.BurstInterval****Property**

**Syntax** `ATS2.AGen.BurstInterval (ByVal Unit As String)`

**Data Type** Double      2 - 65536 cycles or equivalent period based on sine waveform frequency.

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Cycles, sec

**Description** This command sets the number of cycles between the start of a burst and the start of the following burst. This number may be from 2 to 65535 cycles and must be greater than the number of ON cycles. If the number of cycles attempted is not greater than the ON cycles, the interval is not changed.

Note that the interval will occur immediately when this command is called if the burst is running.

**See Also** `ATS2.AGen.Wfm`, `ATS2.AGen.BurstLevel`,  
`ATS2.AGen.BurstOnTime`

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.Wfm.Sine.ShapedBurst
    ATS2.AGen.OutputOn = True
    ATS2.AGen.BurstInterval("Cycles") = 10
    ATS2.AGen.BurstOnTime("Cycles") = 5
    ATS2.AGen.BurstLevel("dB") = -40
    Interval = ATS2.AGen.BurstInterval("Cycles")
    Ontime = ATS2.AGen.BurstOnTime("Cycles")
    Level = ATS2.AGen.BurstLevel("%")
    Debug.Print "Burst Interval =" ; Interval ; " cycles."
    Debug.Print "Burst ON time =" ; Ontime ; " cycles."
    Debug.Print "Burst OFF time low level =" ; Level ;
    %."
End Sub
```

**Output**

```
Burst Interval = 10 cycles.
Burst ON time = 5 cycles.
Burst OFF time low level = 1 %.
```



**ATS2.AGen.BurstLevel****Property**

<b>Syntax</b>	<b>ATS2.AGen.BurstLevel</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	Level of signal during burst off time. (0 - -80.25dB)
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	The following units are available X/Y, dB, %, PPM.
<b>Description</b>	This command sets the amplitude of the Analog Generator during the burst 'OFF' time. This is as a percentage of the 'ON' amplitude and may range from 100.0 percent to .009716280 percent (-80.25 dB).	
<b>See Also</b>	ATS2.AGen.Wfm, ATS2.AGen.BurstInterval, ATS2.AGen.BurstOnTime	
<b>Example</b>	See example for ATS2.AGen.BurstInterval.	

**ATS2.AGen.BurstOnTime****Property**

<b>Syntax</b>	<b>ATS2.AGen.BurstOnTime</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	From 1 to ATS2.AGen.BurstInterval - 1.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Cycles, sec
<b>Description</b>	This command sets the number of cycles for the Analog Generator Burst On Time. This number may be from 1 to 65534 cycles and must be less than the number of interval cycles. If the number of cycles attempted is not less than the interval cycles, the ON time is not changed.	
<b>See Also</b>	ATS2.AGen.Wfm, ATS2.AGen.BurstInterval, ATS2.AGen.BurstLevel	
<b>Example</b>	See example for ATS2.AGen.BurstInterval.	

**ATS2.AGen.ChBTrackA****Property**

<b>Syntax</b>	<b>ATS2.AGen.ChBTrackA</b>
<b>Data Type</b>	Boolean
	<i>True</i> ON, channel B amplitude tracks channel A amplitude.
	<i>False</i> OFF, channel B amplitude independent of channel A.
<b>Description</b>	This command sets channel "B" amplitude to the same amplitude as set for channel "A".
<b>Example</b>	<pre> Sub Main     ATS.Application.NewTest     ATS2.AGen.Output(apbChA) = True     ATS2.AGen.Output(apbChB) = True     ATS2.AGen.Ampl(apbChA, "Vrms") = 1     <b>ATS2.AGen.ChBTrackA</b> = True     ATS2.AGen.OutputOn = True     ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon     ATS2.AnalogIn.Source(apbChB) = apbAnalogInGenMon     ATS.Anlr.LevelSettling(apbChA, 1, .000025, "V", 3,     .03, apbExponential)     ATS.Anlr.LevelSettling(apbChB, 1, .000025, "V", 3,     .03, apbExponential)     ATS2.Inst.Analyzer.LevelTrig(apbChA)     ATS2.Inst.Analyzer.LevelTrig(apbChB)     Do         ReadyA = ATS2.Inst.Analyzer.LevelReady(apbChA)         ReadyB = ATS2.Inst.Analyzer.LevelReady(apbChB)     Loop Until ReadyA &gt; 0 And ReadyB &gt; 0     ReadingA = ATS2.Inst.Analyzer.LevelRdg(apbChA, "V")     ReadingB = ATS2.Inst.Analyzer.LevelRdg(apbChB, "V")     Debug.Print "Level A amplitude = "; _     Format(ReadingA, "#.0000");" V"     Debug.Print "Level B amplitude = "; _     Format(ReadingB, "#.0000");" V" End Sub </pre>
<b>Output</b>	<pre> Level A amplitude = .9970 V Level B amplitude = .9995 V </pre>

**ATS2.AGen.Config****Property****Syntax**        **ATS2.AGen.Config****Data Type**     Constant*apbAGenBal*

Bal XLR

*apbAGenUnBal*

Unbal BNC/XLR

*apbAGenCMTST*

CMTST XLR

**Description**    This command sets both outputs to a balanced or unbalanced configuration.

Note that the output impedance may change between balanced and unbalanced.

It is possible for this command to cause an amplitude error since the maximum allowable amplitude in the unbalanced configurations is half that for the balanced configuration.

This command sets both outputs to a common mode test configuration.

**See Also**        *ATS2.AGen.Impedance***Example**

Sub Main

`ATS.Application.NewTest``ATS2.AGen.Config = apbAGenBal``ATS2.AGen.Impedance = apbAgenLowZ``ATS2.AGen.Ampl(apbChA, "Vrms") = 2.0``ATS2.AGen.OutputOn = True``ATS2.Inst.Analyzer.RangeAuto(apbChB) = False``ATS2.AnalogIn.Range(apbChA, "Vp") = 2.8``ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon``ATS2.Inst.Analyzer.Analyzer.FuncSettling(apbChA, _  
1, .000002, "V", 4, .05, apbExponential)``ATS2.Inst.Analyzer.FuncTrig(apbChA)``Do``Ready = ATS2.Inst.Analyzer.FuncReady(apbChA)`

```

Loop Until Ready > 0
Rdg = ATS2.Inst.Analyzer.FuncRdg(apbChA, "V")
Debug.Print "Channel A Amplitude = ";Format _
    (Rdg, "#.0000");" V"
ATS2.Inst.Analyzer.RangeAuto(apbChA) = True
End Sub

```

**Output**

Channel A Amplitude = 1.9997 V

**ATS2.AGen.DACBandwidth****Property**

**Syntax**            **ATS2.AGen.DACBandwidth**

**Data Type**        Constant

*apbAGenBW\_30K*

The 30k selection chooses a D/A converter sample rate of 65536 Hz, which permits D/A-based signal generation to a maximum frequency of about 30 kHz and provides the lowest-possible distortion from the D/A converters.

*apbAGenBW\_60K*

The 60k selection chooses a sample rate of 131072 Hz, permitting signal generation to frequencies above 60 kHz but with somewhat higher residual noise and distortion than in the 30k position.

**Description**      This command sets the Analog Generator Digital to Analog converter Bandwidth for all waveforms except Arb Wfm, IMD, Square, and Special Pass Thru and Polarity.

**See Also**            *ATS2.AGen.Wfm*

**Example**

```

Sub Main
ATS.Application.NewTest
ATS2.AGen.Wfm.Sine.Normal
ATS2.AGen.Freq("Hz") = 30000.0
ATS2.AGen.DACBandwidth = apbAGenBW_30K
ATS2.AGen.OutputOn = True
ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
ATS2.Inst.Analyzer.FuncMode = apbAnlrTHDNRatio

```

```

ATS2.Inst.Analyzer.FuncSettling(apbChA, 1.0, _
    .0002, "%", 3, .05, apbExponential)
ATS2.Inst.Analyzer.FuncTrig(apbChA)
Do
    Ready = ATS2.Inst.Analyzer.FuncReady(apbChA)
Loop Until Ready > 0
Reading1 = ATS2.Inst.Analyzer.FuncRdg(apbChA, "%")
Debug.Print "DAC THD" & Chr$(43) & "N at 30k BW = _
    ";Format(Reading1, "#.00000");" %"
ATS2.AGen.DACBandwidth = apbAGenBW_60K
ATS2.Inst.Analyzer.FuncTrig(apbChA)
Do
    Ready = ATS2.Inst.Analyzer.FuncReady(apbChA)
Loop Until Ready > 0
Reading1 = ATS2.Inst.Analyzer.FuncRdg(apbChA, "%")
Debug.Print "DAC THD" & Chr$(43) & "N at 60k BW = _
    ";Format(Reading1, "#.00000");" %"
End Sub

```

**Output**

```

DAC THD+N at 30k BW = .00328 %
DAC THD+N at 60k BW = .00162 %

```

**ATS2.AGen.DACSampleRate****Property**

<b>Syntax</b>	<b>ATS2.AGen.DACSampleRate</b>
<b>Data Type</b>	Constant
	<i>apbAGenSR_65536</i> 65536
	<i>apbAGenSR_131072</i> 131072
	<i>apbAGenSR_OS</i> OSR (Output Sample Rate)
	<i>apbAGenSR_ISR</i> ISR (Input Sample Rate)

**Description** This command sets the Digital to Analog converter sample rate for the Arb Wfm waveform selected with the `ATS2.AGen.Wfm` command.

**See Also** `ATS2.AGen.Wfm`

---

## ATS2.AGen.DualAmplRatio

### Property

**Syntax** `ATS2.AGen.DualAmplRatio (ByVal Unit As String)`

**Data Type** Double Valid settings are 0.00001% to 100%

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: %, dB, PPM, X/Y

**Description** This command sets the Analog Generator amplitude ratio to be used with the channel A and B waveforms for the Sine Dual waveform selection.

**See Also** `ATS2.AGen.Freq`

---

## ATS2.AGen.EqAmpl

### Property

**Syntax** `ATS2.AGen.EqAmpl (ByVal Channel As Constant, ByVal Unit As String)`

**Data Type** Double Valid amplitude settings are 0.0 to 100 %FS.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, PPM, Bits, Vrms, Vp, Vpp, dBu, dBV, dBr

**Description** This command sets the Analog Generator post Eq amplitude.

**See Also** `ATS2.AGen.EqCurve`

**Example** `Sub Main`

```

ATS.Application.NewTest
ATS2.AGen.EqCurve("75US-PRE.ATSQ", 1)
ATS2.AGen.Wfm.Sine.EQ
ATS2.AGen.EqAmp1(apbChA, "dBV") = -10.0
ATS2.AGen.OutputOn = True
ATS.Sweep.Data(1).Top("dBV") = 12.0
ATS.Sweep.Data(1).Bottom("dBV") = -12.0
ATS.Sweep.Stereo = True
ATS.Sweep.Start
End Sub

```

---

## ATS2.AGen.EqCurve

## Method

**Syntax** `ATS2.AGen.EqCurve (ByVal FileName As String, ByVal Column As Integer)`

**Data Type** Boolean

Parameter	Name	Description
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters. The file must be an ATS Eq file (.atsq).
	<i>Column</i>	0 = Source 1 settings. 1 = Data 1 measurements. 2 = Data 2 measurements. 3 = Data 3 measurements. 4 = Data 4 measurements. 5 = Data 5 measurements. 6 = Data 6 measurements. 7 = Source 2 settings.

**Description** This command attaches a Eq file to the Analog Generator. Values in the file will be used as multiply factors in calculating the Analog Generator Amplitude value.

**Example** See example for `ATS2.AGen.EqAmp1`.

---

## ATS2.AGen.EqCurveColumn

## Get Only Property

**Syntax** `ATS2.AGen.EqCurveColumn (ByVal Data As Integer)`

<b>Data Type</b>	Integer	Column number.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Data</i>	Number of the Sweep Data (1-6) of the data in memory.
<b>Description</b>	This command returns the column number in the attached file used in the Analog Generator EqCurve waveform selection.	
<b>See Also</b>	ATS2.AGen.EqCurve, ATS2.AGen.EqCurveFilename	

## ATS2.AGen.EqCurveFilename

### Get Only Property

<b>Syntax</b>	<b>ATS2.AGen.EqCurveFilename</b>	
<b>Data Type</b>	Integer	Any valid DOS filename and extension.
<b>Description</b>	This command returns the File Name of the attached file used for the Analog Generator EqCurve waveform selection.	
<b>See Also</b>	ATS2.AGen.EqCurve, ATS2.AGen.EqCurveColumn	

## ATS2.AGen.Freq

### Property

<b>Syntax</b>	<b>ATS2.AGen.Freq</b> (ByVal <i>Channel</i> As String, ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	Valid frequency settings for the Hz unit and sine waveform are 2.0 - 61.665 kHz.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Channel</i>	apbFreq1 = Sets the Frequency (Primary) for all Sine, Square, and Special waveform selections. apbFreq2 = Sets Frequency 2 (Secondary) for the Sine Stereo and Dual waveforms.
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM
<b>Description</b>	This command sets the Analog Generator Frequency.	
<b>Example</b>	Sub Main	



```

ATS.Application.NewTest
ATS2.AGen.Wfm.sine.Stereo
ATS2.AGen.Outputs = True
ATS2.AGen.Freq(apbFreq1, "Hz") = 1000.0
ATS2.AGen.Freq(apbFreq2, "Hz") = 2000.0
ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
ATS2.AnalogIn.Source(apbChB) = apbAnalogInGenMon
ATS2.Inst.Analyzer.FreqSettling(apbChA, .5, _
    .0002, "Hz", 3, .03, apbExponential)
ATS2.Inst.Analyzer.FreqSettling(apbChB, .5, _
    .0002, "Hz", 3, .03, apbExponential)
ATS2.Inst.Analyzer.FreqTrig(apbChA)
ATS2.Inst.Analyzer.FreqTrig(apbChB)
Do
    AReady = ATS2.Inst.Analyzer.FreqReady(apbChA)
    BReady = ATS2.Inst.Analyzer.FreqReady(apbChB)
Loop Until AReady > 0 And BReady > 0
AReading = ATS2.Inst.Analyzer.FreqRdg(apbChA, "Hz")
BReading = ATS2.Inst.Analyzer.FreqRdg(apbChB, "Hz")
Debug.Print "Channel A Frequency = " & _
    Format(AReading, "#.00") & " Hz"
Debug.Print "Channel B Frequency = " & _
    Format(BReading, "#.00") & " Hz"
End Sub

```

**Output**

```

Channel A Frequency = 1000.00 Hz
Channel B Frequency = 2000.00 Hz

```

**ATS2.AGen.IMFreq****Property**

<b>Syntax</b>	<b>ATS2.AGen.IMFreq</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	Frequencies from 40 Hz to 500 Hz are available.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command. Hz
<b>Description</b>	This command sets the Analog Generator IMD lower frequency tone.	

Set the generator to an IMD waveform before calling this command in order to have the proper IMD frequency selected.

**See Also**

ATS2.AGen.Wfm, ATS2.AGen.IMHighFreq

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.Wfm.IMD.SMPTE
    ATS2.AGen.IMHighFreq("Hz") = 7000
    ATS2.AGen.IMFreq("Hz") = 60
    ATS2.AGen.Ampl("dBu") = 0.0
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.Inst.Analyzer.FuncMode = apbAnlrSMPTE
    ATS2.Inst.Analyzer.FuncSettling(apbChA, 3, _
        0.00003, "%", 3, .05, apbExponential)
    ATS2.Inst.Analyzer.FuncTrig(apbChA)
    Do
        Ready = ATS2.Inst.Analyzer.FuncReady(apbChA)
        Loop Until Ready > 0
    Reading1 = ATS2.Inst.Analyzer.FuncRdg(apbChA, "%")
    Debug.Print "SMPTE 4:1 = " & Format(Reading1, _
        "#.0000") & " %"
End Sub
```

**Output**

SMPTE 4:1 = .0010 %

**ATS2.AGen.IMHighFreq****Property**

**Syntax** **ATS2.AGen.IMHighFreq**(ByVal *Unit* As String)

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command. Hz

**Description** This command sets the Analog Generator IMD High Frequency. The frequency passed is rounded to the closest available value.

Set the Analog Generator waveform to an IMD before calling this command.

**See Also** `ATS2.AGen.IMFreq`

**Example** See example for `ATS2.AGen.IMFreq`.

---

## ATS2.AGen.Impedance

**Property**

**Syntax** `ATS2.AGen.Impedance`

**Data Type** Integer

The following list contains the selections relevant to the `ATS2.AGen.Config` command for the Balanced and CMTST selections.

`apbAgenLowZ`

40

`apbAgenHighZ`

150 Ohms standard, 200 Ohms for EURZ option, 600 Ohms for 600Z option.

The following list contains the selections relevant to the `ATS2.AGen.Config` command for the Un-Balanced selections.

`apbAgenLowZ`

20

`apbAgenHighZ`

50

**Description** This command controls the output impedance for Balanced and Un-Balanced generator output configurations.

**See Also** `ATS2.AGen.Config`

---

## ATS2.AGen.Output

**Property**

**Syntax** `ATS2.AGen.Output (ByVal Channel As Constant)`

**Data Type** Boolean

`True` On

	<i>False</i>	Off
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
<b>Description</b>	This command sets the Analog Generator specified channel output to ON or OFF.	
<b>See Also</b>	ATS2.AGen.OutputOn	
<b>Example</b>	See example for ATS2.AGen.Ampl.	

---

## ATS2.AGen.OutputOn

## Property

<b>Syntax</b>	<b>ATS2.AGen.OutputOn</b>	
<b>Data Type</b>	Boolean	
	<i>True</i>	On
	<i>False</i>	Off
<b>Description</b>	This command sets the Analog Generator channel A and B outputs to ON or OFF if they have been individually enabled by the ATS2.AGen.Output command.	
<b>See Also</b>	ATS2.AGen.Output	
<b>Example</b>	See example for ATS2.AGen.Ampl.	

---

## ATS2.AGen.Phase

## Property

<b>Syntax</b>	<b>ATS2.AGen.Phase</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: deg
<b>Description</b>	This command sets the Analog Generator Phase value.	

Set the Analog Generator waveform to Sine Var Phase before calling this command.

### Example

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.Wfm.Sine.VarPhase
    ATS2.AGen.Phase("deg") = 90.0
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.AnalogIn.Source(apbChB) = apbAnalogInGenMon
    ATS2.AGen.OutputOn = True|
    ATS2.Inst.Analyzer.FuncMode = apbAnlrPhase
    ATS2.Inst.Analyzer.FuncSettling(apbChA, 0.0, _
        0.2, "deg", 2, 0.02, apbFlat)
    ATS2.Inst.Analyzer.FuncTrig(apbChA)
    Do
        Ready = ATS2.Inst.Analyzer.FuncReady(apbChA)
    Loop Until Ready > 0
    Reading1 = ATS2.Inst.Analyzer.FuncRdg(apbChA, "deg")
    Debug.Print "Channel B is " & Format(Reading1, _
        "##.000") & " deg relative to channel A."
End Sub
```

### Output

Channel B is 89.984 deg relative to channel A.

---

## ATS2.AGen.RefdBm

## Property

**Syntax**      **ATS2.AGen.RefdBm**(ByVal *Unit* As String)

**Data Type**      Double                  Impedance value.

Parameter	Name	Description
	<i>Unit</i>	Ohms only.

**Description**      This command sets the value known to be the generator load impedance for use in dBm computations. When a value of generator output amplitude is requested via the APAGen.Ampl command using the dBm unit, the software uses this dBm reference impedance value as the "R" in the  $V^2/R$  power computation and sets the generator open-circuit voltage commensurately with the voltage division ratio of

the present generator source impedance and the specified load impedance in order to deliver the specified dBm value to the load.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.Output(apbChB) = False
    ATS2.AGen.Impedance = 2
    ATS2.AGen.RefdBm("Ohms") = 600
    ATS2.AGen.Ampl("dBm") = 0.0
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Impedance(apbChA) = 1
    ATS2.Inst.RefdBm("Ohms") = 600
    Reference = ATS2.Inst.RefdBm("Ohms")
    ATS2.Inst.Analyzer.FuncSettling(1, .000002, _
        "V", 4, .05, 1)
    ATS2.Inst.Analyzer.FuncTrig
Do
    Ready = ATS2.Inst.Analyzer.FuncReady
Loop Until Ready > 0
    Reading1 = ATS2.Inst.Analyzer.FuncRdg("dBm")
    Debug.Print "Channel A Amplitude = ";Format _
        (Reading1, "#.0000");" dBm (";Reference; _
        " Ohms )"
End Sub
```

**Output**

```
Channel A Amplitude = -.0063 dBm ( 600 Ohms )
```

**ATS2.AGen.RefdBr****Property**

**Syntax** `ATS2.AGen.RefdBr (ByVal Unit As String)`

**Data Type** Double      Amplitude value.

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: V, dBu, dBV

**Description** This command sets the zero dBr value for the Analog Generator dBr units.

**Example** Sub Main

```

ATS.Application.NewTest
ATS2.AGen.RefdBr("V") = 1
ATS2.AGen.Ampl(apbChA, "dBr") = 1
ATS2.AGen.OutputOn = True
ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
ATS2.Inst.Analyzer.RefdBr(apbChA, "V") = 1
Reference = ATS2.AGen.RefdBr("V")
ATS2.Inst.Analyzer.LevelSettling(apbChA, 1, _
    .000002, "V", 4, .05, apbExponential)
ATS2.Inst.Analyzer.LevelTrig(apbChA)
Do
    Ready = ATS2.Inst.Analyzer.LevelReady(apbChA)
Loop Until (Ready > 0)
Reading1 = ATS2.Inst.Analyzer.LevelRdg(apbChA, _
    "dBr A")
Debug.Print "Channel A Amplitude = ";Format _
    (Reading1, "#.0000");" dBr relative to"; _
    Reference;" Volts"
End Sub

```

**Output**

Channel A Amplitude = .9679 dBr relative to 1 Volts

**ATS2.AGen.RefdBrAuto****Method**

**Syntax**            **ATS2.AGen.RefdBrAuto**

**Result**            Boolean

*True*                dBr reference set.  
*False*                dBr reference not set.

**Description**     This command sets the generator dBr reference field to the current generator Amplitude setting. If the command is successful a boolean True is returned. If the command is not successful a boolean False is returned.

**Example**            Sub Main  
                       ATS.Application.NewTest  
                       ATS2.AGen.Ampl(apbChA, "dBV") = 0  
                       **ATS2.AGen.RefdBrAuto**  
                       'Increase amplitude 2 dB.

```

ATS2.AGen.Ampl(apbChA, "dbr") = 2
ATS2.AGen.OutputOn = True
ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
ATS2.Inst.Analyzer.FuncMode = apbAnlrAmplitude
ATS2.Inst.Analyzer.FuncSettling(apbChA, 1, _
    .000002, "V", 4, .05, apbExponential)
ATS2.Inst.Analyzer.FuncTrig(apbChA)
Do
    Ready = ATS2.Inst.Analyzer.FuncReady(apbChA)
Loop Until Ready > 0
Reading1 = ATS2.Inst.Analyzer.FuncRdg(apbChA,
"dBV")
Debug.Print "Channel A Amplitude =";Format$ _
    (Reading1,"#.000000");" dBV"
End Sub

```

**Output**

```
Channel A Amplitude = 1.974047 dBV
```

**ATS2.AGen.RefFreq****Property**

**Syntax**      **ATS2.AGen.RefFreq**(ByVal *Unit* As String)

**Data Type**      Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command: Hz

**Description**      This command sets the Analog Generator relative frequency reference value. This reference is used for all the Analog Generator relative frequency units (F/R, dHz, %Hz, cent, octs, decs, d%, dPPM)

**See Also**      `ATS2.AGen.Freq`

**Example**

```

Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AGen.RefFreq("Hz") = 5000
    ATS2.AGen.Freq(apbChA, "dHz") = 5000
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.Inst.Analyzer.FreqSettling(apbChA, .5, _

```



```

        .0002, "Hz", 3, .03, apbExponential)
ATS2.Inst.Analyzer.FreqTrig (apbChA)
Do
    Ready = ATS2.Inst.Analyzer.FreqReady (apbChA)
Loop Until Ready > 0
Reading1 = ATS2.Inst.Analyzer.FreqRdg (apbChA, "Hz")
Debug.Print "Frequency A = " & Format (Reading1, _
    "#.0000") & " Hz"
End Sub

```

**Output**                      Frequency A = 9996.5878 Hz

---

## ATS2.AGen.RefFreqAuto

## Method

**Syntax**                      **ATS2.AGen.RefFreqAuto**

**Result**                      Boolean

*True*                          Frequency reference set.  
*False*                         Frequency reference not set.

**Description**                This command sets the generator frequency reference field to the current generator frequency setting. If the command is successful a boolean True is returned. If the command is not successful a boolean False is returned.

**Example**

```

Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AGen.RefFreqAuto
    ATS2.AGen.Freq (apbFreq1, "dHz") = 2000
    ATS2.AnalogIn.Source (apbChA) = apbAnalogInGenMon
    ATS2.Inst.Analyzer.FreqSettling (apbChA, .5, _
        0.0002, "Hz", 3, .03, apbExponential)
    ATS2.Inst.Analyzer.FreqTrig (apbChA)
Do
    Ready = ATS2.Inst.Analyzer.FreqReady (apbChA)
Loop Until Ready > 0
Reading1 = ATS2.Inst.Analyzer.FreqRdg (apbChA, "Hz")
Debug.Print "Frequency A =" & _
    Format$(Reading1, "#.00") & " Hz"

```

```
End Sub
```

**Output**      Frequency A = 3000.00 Hz

## ATS2.AGen.RefWatts

## Property

**Syntax**      **ATS2.AGen.RefWatts** (ByVal *Unit* As String)

**Data Type**      Double      Impedance value.

Parameter	Name	Description
	<i>Unit</i>	Ohms only.

**Description**      This command sets the value known to be the generator load impedance for use in Watts computations. When a value of generator output amplitude is requested via the `ATS2.AGen.Ampl` command using the Watts unit, the software uses this Watts reference impedance value as the "R" in the  $V^2/R$  power computation and sets the generator open-circuit voltage commensurately with the voltage division ratio of the present generator source impedance and the specified load impedance in order to deliver the specified power value to the load.

### Example

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AGen.RefWatts("Ohms") = 8
    ATS2.AGen.Ampl(apbChA, "W") = .1
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.Inst.Analyzer.RefWatts("Ohms") = 8
    ATS2.Inst.Analyzer.FuncSettling(apbChA, 1, _
        .000002, "V", 4, .05, apbExponential)
    ATS.AnlR.FuncTrig(apbChA)
    Do
        Ready = ATS2.Inst.Analyzer.FuncReady(apbChA)
    Loop Until Ready > 0
    Reading1 = ATS2.Inst.Analyzer.FuncRdg(apbChA, "W")
    Debug.Print "Output Power = " & _
        Format$(Reading1, "#.0000") & " Watts"
End Sub
```

**Output**            Output Power = 3.5797 Watts

## ATS2.AGen.Wfm

## Method

**Syntax**            `ATS2 . AGen . Wfm . Primary . Secondary`

Parameter	Name	Description
	<i>Primary</i>	This part defines the basic waveform type.
	<i>Secondary</i>	This part defines the basic waveform modifier.

Primary	Secondary	Description
.Sine	.Normal	Sine Normal
	1	Var Phase
	2	Stereo
	3	Dual
	4	Shaped Burst
	5	EQ Sine
.IMD	.SMPTE_4_1	SMPTE/DIN 4:1
	.SMPTE_1_1	SMPTE/DIN 1:1
.Square		Square
.Noise		Noise
.Arbitrary		Arb Wfm
.Special	.Polarity	Polarity
	.PassThru	Pass Thru

**Description**    This command selects the Analog Generator waveform. The table above shows the possible settings for the `ATS2 . AGen . Wfm` command.

**Example**

```
Sub Main
  ATS.Application.NewTest
  ATS2.AGen.Wfm.Arbitrary
  ATS2.AGen.WfmName = "48kSine996-0dB.agm"
  ATS2.AGen.Ampl(apbChA, "Vrms") = 2.0
  ATS2.AGen.OutputOn = True
```

```

ATS2.AnalogIn.Source (apbChA) = apbAnalogInGenMon
ATS2.AnalogIn.Source (apbChB) = apbAnalogInGenMon
ATS2.AnalogIn.SampleRate = apbAnalogInHIRES_OSR
ATS2.Inst.Selection = apbInstHarmonicAnalyzer

With ATS2.Inst.Harmonic
    .InputFormat = apbInstInAnalog
    .AmplTrig (apbChA)
    .FreqTrig (apbChA)
Do
Loop Until .AmplReady (apbChA) _
    And .FreqReady (apbChA)
var1 = .AmplRdg (apbChA, "V")
var2 = .FreqRdg (apbChA, "Hz")
End With
Text1$= "Channel A Fundamental Amplitude " _
    & Str$(Format (var1, "##.000")) & "V"
Text2$= "Channel A Fundamental Frequency " _
    & Str$(Format (var2, "##.000")) & "Hz"
ATS.Prompt.Text = Text1$ & vbCr & Text2$
ATS.Prompt.ShowWithContinue
End Sub

```

---

## ATS2.AGen.WfmName

## Property

<b>Syntax</b>	<b>ATS2.AGen.WfmName</b>	
<b>Data Type</b>	String	Long Path and File Names permitted up to 128 characters.
<b>Description</b>	This command loads the designated arbitrary waveform file (.AGM or .AGS) into the Analog Generator. The file must be an ATS waveform file (.agm or .ags).	
	<p>Note: If the Digital Generator waveform selection is set to Arbitrary then this command will also change the Arbitrary waveform for the Digital Generator.</p>	
<b>See Also</b>	ATS2.AGen.Wfm	

### ATS2.AnalogIn.DCCoupled

Property

**Syntax** `ATS2.AnalogIn.DCCoupled (ByVal Channel As Constant)`

**Data Type** Boolean

<i>True</i>	DC Coupled
<i>False</i>	Not DC Coupled

**Parameter**

Name	Description
<i>Channel</i>	apbChA = Channel A apbChB = Channel B

**Description** This command sets the specified channel Input Coupling. This enables the ATS-2 to DC couple the input to the Analyzer for improved CMRR at low frequencies and increased low frequency measurement capability. By DC coupling the Analog Input DC Volts can also be measured.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.AnalogIn.Source(apbChB) = apbAnalogInGenMon
    ATS2.AnalogIn.DCCoupled(apbChA) = True
    ATS2.AnalogIn.DCCoupled(apbChB) = True
    ATS2.AnalogIn.PeakTrig(apbChA)
    ATS2.AnalogIn.PeakTrig(apbChB)
Do
    AReady = ATS2.AnalogIn.PeakReady(apbChA)
    BReady = ATS2.AnalogIn.PeakReady(apbChB)
Loop Until AReady > 0 And BReady > 0
AReading = ATS2.AnalogIn.PeakRdg(apbChA, "Vp")
```

```

BReading = ATS2.AnalogIn.PeakRdg(apbChB, "Vp")
Debug.Print "Channel A Peak Level = " _
    & Format(AReading, "#.0000") & " Vp"
Debug.Print "Channel B Peak Level = " _
    & Format(BReading, "#.0000") & " Vp"
End Sub

```

**Output** Channel A Peak Level = 1.414 Vp  
Channel B Peak Level = 1.414 Vp

---

## ATS2.AnalogIn.Impedance

## Property

**Syntax** **ATS2.AnalogIn.Impedance** (ByVal *Channel* As Constant)

**Data Type** Constant

*apbInputLowZ*

100k Ohms standard, 600 Ohms for 600Z option.

*apbInputHighZ*

100k

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B

**Description** This command selects one of the available termination impedances for the Analyzer Input.

**Example** See example for `ATS2.Inst.RefdBm`.

---

## ATS2.AnalogIn.PeakRdg

## Property

**Syntax** **ATS2.AnalogIn.PeakRdg** (ByVal *Channel* As Constant, ByVal *Unit* As String)

**Data Type** Double

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Unit</i>	The following unit is available: Vp
<b>Description</b>	This command returns a unsettled reading for the Analog Input Peak meter and zeros the ready count.	
<b>See Also</b>	ATS2.AnalogIn.PeakReady, ATS2.AnalogIn.PeakTrig	
<b>Example</b>	See example for ATS2.AnalogIn.DCCoupled.	

## ATS2.AnalogIn.PeakReady

## Property

**Syntax**      **ATS2.AnalogIn.PeakReady** (ByVal *Channel* As Constant)

**Data Type**      Integer

0                      Reading not ready.  
>0                     Reading ready.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description**      This command returns the Analog Input Peak meter unsettled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the `ATS2.AnalogIn.PeakRdg` or `ATS2.AnalogIn.PeakTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.AnalogIn.FreqRdg` command will be guaranteed to return quickly.

**See Also**            `ATS2.AnalogIn.PeakRdg`, `ATS2.AnalogIn.PeakTrig`

**Example**            See example for `ATS2.AnalogIn.DCCoupled`.

## ATS2.AnalogIn.PeakTrig

**Method**

**Syntax** `ATS2.AnalogIn.PeakTrig (ByVal Channel As Constant)`

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.AnalogIn.PeakRdg` command. The reading in progress is aborted.

**See Also** `ATS2.AnalogIn.PeakRdg`, `ATS2.AnalogIn.PeakReady`

**Example** See example for `ATS2.AnalogIn.DCCoupled`.

## ATS2.AnalogIn.Range

**Property**

**Syntax** `ATS2.AnalogIn.Range (ByVal Channel As Constant, ByVal Unit As String)`

**Data Type** Double The following values are the range boundaries for the “dBV” unit: 42.095, 36.075, 30.054, 23.995, 17.974, 11.953, 5.933, -0.088, -5.985, -12.006

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>Unit</i>	The following units are available: Vp, dBu, dBV

**Description** This command sets the Analog Input Range and returns the nominal full scale of the range in use.

**See Also** `ATS2.AnalogIn.RangeAuto`

## ATS2.AnalogIn.RangeAuto

**Property**

**Syntax** `ATS2.AnalogIn.RangeAuto (ByVal Channel As Constant)`



**Data Type** Boolean

*True* Auto range  
*False* Fixed range

**Parameter**

Name	Description
<i>Channel</i>	apbChA = Channel A apbChB = Channel B

**Description**

This command sets the Analog Input to Auto range or Fixed range. Care must be taken when using Fixed range that the input signal does not exceed the selected range.

**See Also**

ATS2.AnalogIn.Range

---

## ATS2.AnalogIn.SampleRate

**Property**

**Syntax** `ATS2.AnalogIn.SampleRate`

**Data Type**

Constant

*apbAnalogInHIRES\_65536*  
HiRes@65536

*apbAnalogInHIRES\_OS*  
HiRes@OSR

*apbAnalogInHIBW\_131072*  
HiBW@131072

*apbAnalogInHIBW\_262144*  
HiBW@262144

*apbAnalogInHIBW\_2XOSR*  
HiBW@2xOSR

**Description**

This command sets the Analog Input Converter Sample Rate.

---

## ATS2.AnalogIn.Source

**Property**

**Syntax**

`ATS2.AnalogIn.Source (ByVal Channel As Constant,  
ByVal Input as Integer)`

<b>Data Type</b>	Constant				
	<i>apbAnalogInXLR_Bal</i> XLR (Bal): Front panel XLR Analog Input connector, balanced				
	<i>apbAnalogInBNC_Unbal</i> BNC (unbal): Front panel BNC Analog Input connector, unbalanced				
	<i>apbAnalogInGenMon</i> GenMon: Internal connection from Analog Generator to Analog Input				
<b>Parameter</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>Channel</i></td> <td>apbChA = Channel A apbChB = Channel B</td> </tr> </tbody> </table>	Name	Description	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
Name	Description				
<i>Channel</i>	apbChA = Channel A apbChB = Channel B				
<b>Description</b>	This command selects the Analog Input connection for the specified channel.				
<b>Example</b>	See example for <code>ATS2.AnalogIn.DCCoupled</code> .				

### ATS2.AuxControlIO.InputBitRdg

Property

**Syntax** `ATS2.AuxControlIO.InputBitRdg (ByVal Unit As String)`

**Data Type** Boolean

*True* Bit state one.  
*False* Bit state zero.

**Parameter**

Name	Description
<i>Bit</i>	Bit value 0 - 7 allowed.

**Description** This command returns the state of a single bit of the Auxiliary Input.

**See Also** `ATS2.AuxControlIO.Output`

**Example**

```
Sub Main
    ATS.Application.PanelOpen (apbAuxDigIO)
    ATS2.AuxControlIO.Output("dec") = 85
    Wait .5
    For Bit = 0 To 7
        Debug.Print " & Bit & " set " & _
            CBool(ATS2.AuxControlIO.InputBitRdg(Bit))
    Next Bit
End Sub
```

**Output**

```
0 set True
1 set False
2 set True
3 set False
4 set True
5 set False
6 set True
7 set False
```

## ATS2.AuxControlIO.InputRdg Property

**Syntax** `ATS2.AuxControlIO.InputRdg (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	The following units are available: dec, hex, and oct.

**Description** This command returns a reading for the Auxiliary Input.

**See Also** `ATS2.AuxControlIO.InputBitRdg`

**Example**

```
Sub Main
    ATS.Application.PanelOpen (apbAuxDigIO)
    ATS2.AuxControlIO.Output("dec") = 16
    Wait .5
    Debug.Print ATS2.AuxControlIO.InputRdg("dec")
End Sub
```

**Output** 16

## ATS2.AuxControlIO.Output Property

**Syntax** `ATS2.AuxControlIO.Output (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Units\$</i>	The following units are available dec, hex, oct, h(x).

**Description** This command sets the value of the Auxiliary Output.

**See Also** `ATS2.AuxControlIO.OutputBit`

**Example**

```
Sub Main
    ATS.Application.PanelOpen (apbAuxDigIO)
    ATS2.AuxControlIO.Output("dec") = 16
    ATS2.AuxControlIO.Output("hex") = &h22
    ATS2.AuxControlIO.Output("oct") = &o104

    ATS2.AuxControlIO.Output("dec") = 0
    ATS2.AuxControlIO.OutputBit(0) = True
```

End Sub

---

## ATS2.AuxControlIO.OutputBit

**Property**

**Syntax**            **ATS2.AuxControlIO.OutputBit** (ByVal *Bit* As Integer)

**Data Type**        Boolean

*True*                Set bit state to one.  
*False*               Set bit state to zero.

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Bit</i>	Bit value 0 - 7 allowed.

**Description**     This command sets the state of a single bit of the Auxiliary Output.

**See Also**         `ATS2.AuxControlIO.Output`

**Example**         See example for `ATS2.AuxControlIO.Output`.



### ATS2.Bits.AudioModeRdg

Property

**Syntax** `ATS2.Bits.AudioModeRdg (ByVal Channel As Constant, [Optional ByVal String As Variant])`

**Data Type** Integer

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

Result	Constant	Description
	<i>apbBitsConAudio</i>	Consumer Audio Mode
	<i>apbBitsConData</i>	Consumer Data Mode
	<i>apbBitsProNonAudio</i>	Professional Non Audio Mode
	<i>apbBitsProNormal</i>	Professional Normal Mode

**Description** This command returns the Status Bits Audio Mode from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example** Sub Main

```

Debug.Print "Audio Mode = ";
Select Case (ATS2.Bits.AudioModeRdg (apbChA))
    Case apbBitsConAudio
        Debug.Print "Consumer Audio Mode"
    Case apbBitsConData
        Debug.Print "Consumer Data Mode"
    Case apbBitsProNonAudio
        Debug.Print "Professional Non Audio Mode"
    Case apbBitsProNormal
        Debug.Print "Professional Normal Mode"
End Select
End Sub

```

## ATS2.Bits.AuxBitsRdg

## Property

**Syntax**      **ATS2.Bits.AuxBitsRdg** (ByVal *Channel* As Constant, [Optional ByVal *String* As Variant])

**Data Type**      Integer

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

Result	Constant	Description
	<i>apbBitsPro20BitNotDefined</i>	20-bit not defined
	<i>apbBitsPro24BitMain</i>	24-bit not defined
	<i>apbBitsPro20BitSingle</i>	20-bit single
	<i>apbBitsProBitsReserved</i>	Reserved

**Description**      This command returns the Status Bits Auxiliary Bits from an optional string or from the AES/EBU data stream. When the optional string



parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

### Example

```
Sub Main
  If ATS2.Bits.ModeRdg(apbChA) = apbBitsModePro Then
    Debug.Print "Auxiliary Bits Reading = ";
    Select Case (ATS2.Bits.AuxBitsRdg(apbChA))
      Case apbBitsPro20BitNotDefined
        Debug.Print "20-bit not defined"
      Case apbBitsPro24BitMain
        Debug.Print "24-bit main audio"
      Case apbBitsPro20BitSingle
        Debug.Print "20-bit single"
      Case apbBitsProBitsReserved
        Debug.Print "Reserved"
    End Select
  End If
End Sub
```

## ATS2.Bits.CategoryRdg

## Property

**Syntax** `ATS2.Bits.CategoryRdg (ByVal Channel As Constant, [Optional ByVal String As Variant])`

**Data Type** Integer

### Parameter

Name	Description
<i>Channel</i>	apbChA = Channel A apbChB = Channel B
<i>String</i>	Optional string containing status bit information.

### Result

Constant	Description
<i>apbBitsConGeneral</i>	General

```

    apbBitsConCD
        CD Player
    apbBitsConPCM
        PCM Adaptor
    apbBitsConDAT
        DAT Recorder
    apbBitsConDigBroadcast
        Digital Broadcast
    apbBitsConMusInst
        Musical Instrument
    apbBitsConDVD
        DVD
    apbBitsConPresAD
        Present A/D Converter
    apbBitsConFutureAD
        Future A/D Converter
    apbBitsConSolidState
        Solid State Memory
    apbBitsConExperimental
        Experimental

```

**Description** This command returns the Status Bits Category code from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```

Sub Main
    If ATS2.Bits.ModeRdg(apbChA) = apbBitsModeCons Then
        Debug.Print "Category Reading = ";
        Select Case (ATS2.Bits.CategoryRdg(apbChA))
            Case apbBitsConGeneral
                Debug.Print "General"
            Case apbBitsConCD

```

```

        Debug.Print "CD Player"
    Case apbBitsConPCM
        Debug.Print "PCM Adaptor"
    Case apbBitsConDAT
        Debug.Print "DAT Recorder"
    Case apbBitsConDigBroadcast
        Debug.Print "Digital Broadcast"
    Case apbBitsConMusInst
        Debug.Print "Musical Instrument"
    End Select
End If
End Sub

```

## ATS2.Bits.ChModeRdg

## Property

**Syntax**      **ATS2.Bits.ChModeRdg** (ByVal *Channel* As Constant, [Optional ByVal *String* As Variant])

**Data Type**    Integer

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

Result	Constant	Description
	<i>apbBitsProNotIndicated</i>	Not Indicated
	<i>apbBitsProTwo</i>	2-channel
	<i>apbBitsProSingle</i>	Single-channel
	<i>apbBitsProPrimSec</i>	Primary/Sec
	<i>apbBitsProStereo</i>	Stereo

```

    apbBitsProReserved1
        Reserved-1
    apbBitsProReserved2
        Reserved-2
    apbBitsProVector
        Vector to byte 3
    apbBitsProMonoDouble
        Mono Double Rate
    apbBitsProLeftDouble
        Left Double Rate
    apbBitsProRightDouble
        Right Double Rate

```

**Description** This command returns the Status Bits Channel Mode from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```

Sub Main
    If ATS2.Bits.ModeRdg(apbChA) = apbBitsModePro Then
        Debug.Print "Channel Mode Reading = ";
        Select Case (ATS2.Bits.ChModeRdg(apbChA))
            Case apbBitsProNotIndicated
                Debug.Print "Not Indicated"
            Case apbBitsProTwo
                Debug.Print "2-channel"
            Case apbBitsProSingle
                Debug.Print "Single-channel"
            Case apbBitsProPrimSec
                Debug.Print "Primary/Sec"
            Case apbBitsProStereo
                Debug.Print "Stereo"
            Case apbBitsProReserved1
                Debug.Print "Reserved-1"
            Case apbBitsProReserved2
                Debug.Print "Reserved-2"

```

```

        Case apbBitsProVector
            Debug.Print "Vector to byte 3"
        Case apbBitsProMonoDouble
            Debug.Print "Mono Double Rate"
        Case apbBitsProLeftDouble
            Debug.Print "Left Double Rate"
        Case apbBitsProRightDouble
            Debug.Print "Right Double Rate"
    End Select
End If
End Sub

```

**Output** Channel Mode Reading = Not Indicated

## ATS2.Bits.ChNumRdg

## Property

**Syntax** **ATS2.Bits.ChNumRdg** (ByVal *Channel* As Constant, [Optional ByVal *String* As Variant])

**Data Type** Integer

### Parameter

Name	Description
<i>Channel</i>	apbChA = Channel A apbChB = Channel B
<i>String</i>	Optional string containing status bit information.

### Result

Constant	Description
<i>apbBitsConCh_DontCare</i>	Don't Care
<i>apbBitsConCh_A</i>	A (Left)
<i>apbBitsConCh_B</i>	B (Right)
<i>apbBitsConCh_C</i>	C
<i>apbBitsConCh_D</i>	D

```

    apbBitsConCh_E
        E
    apbBitsConCh_F
        F
    apbBitsConCh_G
        G
    apbBitsConCh_H
        H
    apbBitsConCh_I
        I
    apbBitsConCh_J
        J
    apbBitsConCh_K
        K
    apbBitsConCh_L
        L
    apbBitsConCh_M
        M
    apbBitsConCh_N
        N
    apbBitsConCh_O
        O

```

**Description** This command returns the Status Bits Channel Number from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```

Sub Main
    If ATS2.Bits.ModeRdg(apbChA) = apbBitsModeCons Then
        Debug.Print "Channel Number Reading = ";
        Select Case (ATS2.Bits.ChNumRdg(apbChA))
            Case apbBitsConCh_DontCare

```

```
        Debug.Print "Don't Care"
    Case apbBitsConCh_A
        Debug.Print "A (Left) "
    Case apbBitsConCh_B
        Debug.Print "B (Right) "
    Case apbBitsConCh_C
        Debug.Print "C"
    Case apbBitsConCh_D
        Debug.Print "D"
    Case apbBitsConCh_E
        Debug.Print "E"
    Case apbBitsConCh_F
        Debug.Print "F"
    Case apbBitsConCh_G
        Debug.Print "G"
    Case apbBitsConCh_H
        Debug.Print "H"
    Case apbBitsConCh_I
        Debug.Print "I"
    Case apbBitsConCh_J
        Debug.Print "J"
    Case apbBitsConCh_K
        Debug.Print "K"
    Case apbBitsConCh_L
        Debug.Print "L"
    Case apbBitsConCh_M
        Debug.Print "M"
    Case apbBitsConCh_N
        Debug.Print "N"
    Case apbBitsConCh_O
        Debug.Print "O"
End Select
End If
End Sub
```

---

## ATS2.Bits.ClockAccuracyRdg

## Property

**Syntax**      **ATS2.Bits.ClockAccuracyRdg** (ByVal *Channel* As  
Constant, [Optional ByVal *String* As Variant])

**Data Type** Integer

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

<b>Result</b>	<b>Constant</b>	<b>Description</b>
	<i>apbBitsConLevel1</i>	Level 1
	<i>apbBitsConLevel2</i>	Level 2
	<i>apbBitsConLevel3</i>	Level 3
	<i>apbBitsConReserved</i>	Reserved

**Description** This command returns the Status Bits Clock Accuracy from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```
Sub Main
  If ATS2.Bits.ModeRdg(apbChA) = apbBitsModeCons Then
    Debug.Print "Clock Accuracy Reading = ";
    Select Case ATS2.Bits.ClockAccuracyRdg(apbChA)
      Case apbBitsConLevel1
        Debug.Print "Level 1"
      Case apbBitsConLevel2
        Debug.Print "Level 2"
      Case apbBitsConLevel3
        Debug.Print "Level 3"
      Case apbBitsConReserved
        Debug.Print "Reserved"
    End Select
  End If
```



End Sub

---

## ATS2.Bits.Cons.AudioMode

**Property**

**Syntax**            **ATS2.Bits.Cons.AudioMode**

**Data Type**        Constant

*apbBitsConAudio*  
Audio Mode

*apbBitsConData*  
Data Mode

**Description**     This command sets the Mode parameter encoded in the Consumer Status Bits.

**Example**            See example for `ATS2.Bits.Cons.Category`.

---

## ATS2.Bits.Cons.Category

**Property**

**Syntax**            **ATS2.Bits.Cons.Category**

**Data Type**        Constant

*apbBitsConGeneral*  
General

*apbBitsConCD*  
CD Player

*apbBitsConPCM*  
PCM Adaptor

*apbBitsConDAT*  
DAT Recorder

*apbBitsConDigBroadcast*  
Digital Broadcast

*apbBitsConMusInst*  
Musical Instrument

*apbBitsConDVD*  
DVD

*apbBitsConPresAD*  
Present A/D Converter

*apbBitsConFutureAD*  
Future A/D Converter

*apbBitsConSolidState*  
Solid State Memory

*apbBitsConExperimental*  
Experimental

**Description** This command sets the Category Code parameter (channel status bit C) encoded in the Consumer Status Bits.

System One digital I/O units always sends the same status bits on Channels A and B.

**Example**

```
Sub Main
  'other setup code ...
  ATS2.Bits.XmitChannel = apbBitsXmitChAB
  ATS2.Bits.Mode = apbBitsModeCons
  ATS2.Bits.Cons.AudioMode = apbBitsConData
  ATS2.Bits.Cons.CopyRight = apbBitsConNonCopyright
  ATS2.Bits.Cons.Emphasis = apbBitsConEmp50_15
  ATS2.Bits.Cons.Channels = apbBitsCon2Channel
  ATS2.Bits.Cons.Category = apbBitsConCD
  ATS2.Bits.Cons.SourceNum = apbBitsCon_1
  ATS2.Bits.Cons.ChNum = apbBitsConCh_A
  ATS2.Bits.Cons.SampleFreq = apbBitsCon_48k
  ATS2.Bits.Cons.ClockAccuracy = apbBitsConLevel1
  'rest of program ...
End Sub
```

---

**ATS2.Bits.Cons.Channels****Property**

<b>Syntax</b>	<b>ATS2.Bits.Cons.Channels</b>
<b>Data Type</b>	Constant <i>apbBitsCon2Channel</i> 2 Channel <i>apbBitsCon4Channel</i> 4 Channel
<b>Description</b>	This command sets the Channel Mode parameter encoded in the Consumer Status Bits.
<b>Example</b>	See example for <code>ATS2.Bits.Cons.Category</code> .

---

**ATS2.Bits.Cons.ChNum****Property**

<b>Syntax</b>	<b>ATS2.Bits.Cons.ChNum</b>
<b>Data Type</b>	Constant <i>apbBitsConCh_DontCare</i> Don't Care <i>apbBitsConCh_A</i> A (Left) <i>apbBitsConCh_B</i> B (Right) <i>apbBitsConCh_C</i> C <i>apbBitsConCh_D</i> D <i>apbBitsConCh_E</i> E <i>apbBitsConCh_F</i> F

```

apbBitsConCh_G
G
apbBitsConCh_H
H
apbBitsConCh_I
I
apbBitsConCh_J
J
apbBitsConCh_K
K
apbBitsConCh_L
L
apbBitsConCh_M
M
apbBitsConCh_N
N
apbBitsConCh_O
O

```

**Description** This command sets the Source Number parameter encoded in the Consumer Status Bits.

**See Also** `ATS2.Bits.XmitChannel`

**Example** See example for `ATS2.Bits.Cons.Category`.

---

## ATS2.Bits.Cons.ClockAccuracy

## Property

**Syntax** `ATS2.Bits.Cons.ClockAccuracy`

**Data Type** Constant

```

apbBitsConsLevel1
Level 1
apbBitsConsLevel2
Level 2

```

*apbBitsConsLevel3*

Level 3

*apbBitsConReserved*

Reserved

**Description** This command sets the Clock Accuracy parameter encoded in the Consumer Status Bits.

**Example** See example for `ATS2.Bits.Cons.Category`.

## ATS2.Bits.Cons.Copyright

## Property

**Syntax** `ATS2.Bits.Cons.Copyright`

**Data Type** Constant

*apbBitsConCopyright*

Copyright

*apbBitsConNonCopyright*

Non-Copyright

**Description** This command sets the Copyright parameter encoded in the Consumer Status Bits.

**Example** See example for `ATS2.Bits.Cons.Category`.

## ATS2.Bits.Cons.Emphasis

## Property

**Syntax** `ATS2.Bits.Cons.Emphasis`

**Data Type** Constant

*apbBitsConEmphNo*

No Pre-emph

*apbBitsConEmph50\_15*

50/15S

**Description** This command sets the Emphasis parameter encoded in the Consumer Status Bits.

**Example** See example for `ATS2.Bits.Cons.Category`.

---

## ATS2.Bits.Cons.SampleFreq Property

**Syntax** `ATS2.Bits.Cons.SampleFreq`

**Data Type** Constant

`apbBitsConFreq_48k`

48 kHz

`apbBitsConFreq_41000`

44.1 kHz

`apbBitsConFreq_32k`

32 kHz

`apbBitsConFreq_24k`

24 kHz

`apbBitsConFreq_22k`

22.05 kHz

`apbBitsConFreq_192k`

192 kHz

`apbBitsConFreq_176k`

176.4 kHz

`apbBitsConFreq_96k`

96 kHz

`apbBitsConFreq_882k`

88.2 kHz

**Description** This command sets the Frequency parameter encoded in the Consumer Status Bits.

**Example** See example for `ATS2.Bits.Cons.Category`.

---

## ATS2.Bits.Cons.SourceNum Property

**Syntax** `ATS2.Bits.Cons.SourceNum`

<b>Data Type</b>	Constant
	<i>apbBitsCon_DontCare</i> Don't Care
	<i>apbBitsCon_1</i> 1
	<i>apbBitsCon_2</i> 2
	<i>apbBitsCon_3</i> 3
	<i>apbBitsCon_4</i> 4
	<i>apbBitsCon_5</i> 5
	<i>apbBitsCon_6</i> 6
	<i>apbBitsCon_7</i> 7
	<i>apbBitsCon_8</i> 8
	<i>apbBitsCon_9</i> 9
	<i>apbBitsCon_10</i> 10
	<i>apbBitsCon_11</i> 11
	<i>apbBitsCon_12</i> 12
	<i>apbBitsCon_13</i> 13
	<i>apbBitsCon_14</i> 14

```
apbBitsCon_15
    15
```

**Description** This command sets the Source Number parameter encoded in the Consumer Status Bits.

**Example** See example for `ATS2.Bits.Cons.Category`.

## ATS2.Bits.CopyrightRdg

## Property

**Syntax** `ATS2.Bits.CopyrightRdg (ByVal Channel As Constant, [Optional ByVal String As Variant])`

**Data Type** Integer

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

Result	Constant	Description
	<i>apbBitsConCopyright</i>	Copyright
	<i>apbBitsConNonCopyright</i>	Non-Copyright

**Description** This command returns the Status Bits Copyright status from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```
Sub Main
    If ATS2.Bits.ModeRdg(apbChA) = apbBitsModeCons Then
        Debug.Print "Copyright Reading = ";
        Select Case (ATS2.Bits.CopyrightRdg(apbChA))
```



```

        Case apbBitsConCopyright
            Debug.Print "Copyright"
        Case apbBitsConNonCopyright
            Debug.Print "Non-Copyright"
    End Select
End If
End Sub

```

## ATS2.Bits.CrcRdg

## Property

**Syntax**      **ATS2.Bits.CrcRdg**(ByVal *Channel* As Constant,  
[Optional ByVal *String* As Variant])

**Data Type**    Integer

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

Result	Value	Description
	<i>True</i>	Valid
	<i>False</i>	Invalid

**Description**    This command returns the Status Bits CRC state from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also**        `ATS2.Bits.StatusXferToString`

**Example**        Sub Main  
                   If ATS2.Bits.ModeRdg(apbChA) = apbBitsModePro Then  
                   Debug.Print "Crc Valid Reading = ";  
                   Select Case (**ATS2.Bits.CrcRdg**(apbChA))  
                   Case True

```

        Debug.Print "Valid"
    Case False
        Debug.Print "Invalid"
    End Select
End If
End Sub

```

## ATS2.Bits.DestinationRdg

## Property

**Syntax**      **ATS2.Bits.DestinationRdg** (ByVal *Channel* As Constant, [Optional ByVal *String* As Variant])

**Data Type**      String

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

**Description**      This command returns the Status Bits Destination Code from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also**      `ATS2.Bits.StatusXferToString`

**Example**

```

Sub Main
    If ATS2.Bits.ModeRdg(apbChA) = apbBitsModePro Then
        Debug.Print "Channel A Destination Reading = " _
            & ATS2.Bits.DestinationRdg(apbChA)
    End If
    If ATS2.Bits.ModeRdg(apbChB) = apbBitsModePro Then
        Debug.Print "Channel B Destination Reading = " _
            & ATS2.Bits.DestinationRdg(apbChB)
    End If
End Sub

```

**ATS2.Bits.EmphRdg****Property**

**Syntax**      **ATS2.Bits.EmphRdg** (ByVal *Channel* As Constant, [Optional ByVal *String* As Variant])

**Data Type**      Integer

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

<b>Result</b>	<b>Constant</b>	<b>Description</b>
---------------	-----------------	--------------------

The following list is for Consumer Mode.

*apbBitsConEmpNo*  
No Pre-emph

*apbBitsConEmp50\_15*  
50/15S

The following list is for Professional Mode.

*apbBitsProEmpNotIndicated*  
Not Indicated

*apbBitsProEmpNone*  
None

*apbBitsProEmp5015*  
50/15 uS

*apbBitsProEmpJ17*  
CCITT J.17

**Description**      This command returns the Status Bits Emphasis setting from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also**      `ATS2.Bits.StatusXferToString`, `ATS2.Bits.ModeRdg`

**Example**

```

Sub Main
    Debug.Print "Emphasis Reading = ";
    Select Case (ATS2.Bits.EmphRdg (apbChA))
        Case apbBitsConEmpNo
            Debug.Print "Consumer No Pre-emph"
        Case apbBitsConEmp50_15
            Debug.Print "Consumer 50/15S"
        Case apbBitsProEmpNotIndicated
            Debug.Print "Professional Not Indicated"
        Case apbBitsProEmpNone
            Debug.Print "Professional None"
        Case apbBitsProEmp5015
            Debug.Print "Professional 50/15S"
        Case apbBitsProEmpJ17
            Debug.Print "Professional CCITT J.17"
    End Select
End Sub

```

**ATS2.Bits.FlagRdg****Property****Syntax**

**ATS2.Bits.FlagRdg** (ByVal *Channel* As Constant, ByVal *FlagId* As Constant, [Optional ByVal *String* As Variant])

**Data Type**

Boolean

**Parameter**

Name	Description
<i>Channel</i>	apbChA = Channel A apbChB = Channel B
<i>FlagId</i>	apbFlag0-5 = Select flag representing bits 0 - 5. apbFlag6-13 = Select flag representing bits 6 - 13. apbFlag14-17 = Select flag representing bits 14 - 17. apbFlag18-21 = Select flag representing bits 18 - 21.
<i>String</i>	Optional string containing status bit information.

**Result**

Value	Description
<i>True</i>	Set
<i>False</i>	Clear

**Description** This command returns the Status Bits Flag state from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```
Sub Main
    Dim String_Array(1)
    String_Array(0) = "Cleared"
    String_Array(1) = "Set"

    ATS.Application.NewTest    'Reset panels
    ATS2.Bits.Mode = 1        'Professional Mode
    ATS2.Dio.InFormat = 3

    'Set flags 14-17
    ATS2.Bits.Pro.Flag(apbFlag14-17) = 1
    'Set Flags 18-21
    ATS2.Bits.Pro.Flag(apbFlag18-21) = 1

    Wait .5 'Wait for reading to update
    Debug.Print "Reliability Flags 0-5 Reading = " & _
        String_Array(ATS2.Bits.FlagRdg _
            (apbChA, apbFlag0-5))
    Debug.Print "Reliability Flags 6-13 Reading = " & _
        String_Array(ATS2.Bits.FlagRdg _
            (apbChA, apbFlag6-13))
    Debug.Print "Reliability Flags 14-17 Reading = " _
        & String_Array(ATS2.Bits.FlagRdg _
            (apbChA, apbFlag14-17))
    Debug.Print "Reliability Flags 18-21 Reading = " _
        & String_Array(ATS2.Bits.FlagRdg _
            (apbChA, apbFlag18-21))
End Sub
```

**Output**

```
Reliability Flags 0-5 Reading = Cleared
Reliability Flags 6-13 Reading = Cleared
Reliability Flags 14-17 Reading = Set
Reliability Flags 18-21 Reading = Set
```

**ATS2.Bits.FreqModeRdg****Property**

**Syntax**      **ATS2.Bits.FreqModeRdg** (ByVal *Channel* As Constant, [Optional ByVal *String* As Variant])

**Data Type**    Integer

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

<b>Result</b>	<b>Constant</b>	<b>Description</b>
	<i>apbBitsProFreqUnLock</i>	Unlocked
	<i>apbBitsProFreqLock</i>	Locked

**Description**    This command returns the Status Bits Frequency Mode from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also**        `ATS2.Bits.StatusXferToString`

**Example**

```
Sub Main
    If ATS2.Bits.ModeRdg(apbChA) = apbBitsModePro Then
        Debug.Print "Frequency Mode Reading = ";
        Select Case (ATS2.Bits.FreqModeRdg(apbChA))
            Case apbBitsProFreqUnLock
                Debug.Print "Unlocked"
            Case apbBitsProFreqLock
                Debug.Print "Locked"
        End Select
    End If
End Sub
```

---

## ATS2.Bits.LocalAddressRdg Property

**Syntax**      **ATS2.Bits.LocalAddressRdg** (ByVal *Channel* As Constant, [Optional ByVal *String* As Variant])

**Data Type**      Long

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

**Description**      This command returns the Status Bits Local Address code from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also**      `ATS2.Bits.StatusXferToString`

**Example**

```
Sub Main
    If ATS2.Bits.ModeRdg(apbChA) = apbBitsModePro Then
        Debug.Print "Ch A Local Address Reading = " _
            & ATS2.Bits.LocalAddressRdg(apbChA)
    End If
End Sub
```

---

## ATS2.Bits.Mode Property

**Syntax**      **ATS2.Bits.Mode**

**Data Type**      Constant

*apbBitsModeCons*  
Consumer

*apbBitsModePro*  
Professional

**Description**      This command sets the Transmit Mode.

**Example** See example for `ATS2.Bits.Cons.Category`.

---

## ATS2.Bits.ModeRdg

**Property**

**Syntax** `ATS2.Bits.ModeRdg (ByVal Channel As Constant, [Optional ByVal String As Variant])`

**Data Type** Integer

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

Result	Constant	Description
	<i>apbBitsModeCons</i>	Consumer
	<i>apbBitsModePro</i>	Professional

**Description** This command returns the Status Bits Mode from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example** See example for `ATS2.Bits.LocalAddressRdg`.

---

## ATS2.Bits.OriginRdg

**Property**

**Syntax** `ATS2.Bits.OriginRdg (ByVal Channel As Constant, [Optional ByVal String As Variant])`

**Data Type** String



Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.
<b>Description</b>	This command returns the Status Bits Origin Code from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the <code>ATS2.Bits.StatusXferToString</code> command.	
<b>See Also</b>	<code>ATS2.Bits.StatusXferToString</code>	
<b>Example</b>	<pre> Sub Main   If ATS2.Bits.ModeRdg(apbChA) = apbBitsModePro Then     Debug.Print "Channel A Origin Reading = " _       &amp; <b>ATS2.Bits.OriginRdg</b>(apbChA)   End If   If ATS2.Bits.ModeRdg(apbChB) = apbBitsModePro Then     Debug.Print "Channel B Origin Reading = " _       &amp; <b>ATS2.Bits.OriginRdg</b>(apbChB)   End If End Sub </pre>	

## ATS2.Bits.Pro.AudioMode

## Property

**Syntax** `ATS2.Bits.Pro.AudioMode`

**Data Type** Constant

*apbBitsProNormal*  
Normal

*apbBitsProNonAudio*  
Non Audio

**Description** This command sets the Audio Mode parameter encoded in the Professional Status Bits.

**Example** See example for `ATS2.Bits.Pro.LocalAddress`.

**ATS2.Bits.Pro.AuxBits****Property**

<b>Syntax</b>	<b>ATS2.Bits.Pro.AuxBits</b>
<b>Data Type</b>	Constant
	<i>apbBitsPro20BitNot</i> 20-bit not defined
	<i>apbBitsPro24BitMain</i> 24-bit main audio
	<i>apbBitsPro20BitSingle</i> 20-bit single
	<i>apbBitsProBitsReserved</i> Reserved
<b>Description</b>	This command sets the Aux Bits parameter encoded in the Professional Status Bits.
<b>Example</b>	See example for <code>ATS2.Bits.Pro.LocalAddress</code> .

**ATS2.Bits.Pro.ChMode****Property**

<b>Syntax</b>	<b>ATS2.Bits.Pro.ChMode</b>
<b>Data Type</b>	Constant
	<i>apbBitsProNotIndicated</i> Not Indicated
	<i>apbBitsProTwo</i> 2-channel
	<i>apbBitsProSingle</i> Single-channel
	<i>apbBitsProPrimSec</i> Primary/Sec
	<i>apbBitsProStereo</i> Stereo

*apbBitsProReserved1*  
Reserved-1

*apbBitsProReserved2*  
Reserved-2

*apbBitsProVector*  
Vector to byte 3

*apbBitsProMonoDouble*  
Mono Double Rate

*apbBitsProLeftDouble*  
Left Double Rate

*apbBitsProRightDouble*  
Right Double Rate

**Description** This command sets the Channel Mode parameter encoded in the Professional Status Bits.

**Example** See example for `ATS2.Bits.Pro.LocalAddress`.

---

## ATS2.Bits.Pro.CrcEnable

## Property

**Syntax** `ATS2.Bits.Pro.CrcEnable`

**Result** Boolean

*True*           Set  
*False*          Clear

**Description** This command sets or clears the CRC parameter encoded in the Professional Status Bits.

The AES3 standard defines byte 23 as a CRC byte to assist the receiver in detecting errors in the preceding 23 bytes (0-22) of each channel status block.

**Example** See example for `ATS2.Bits.Pro.LocalAddress`.

---

**ATS2.Bits.Pro.Destination****Property**

<b>Syntax</b>	<code>ATS2.Bits.Pro.Destination</code>
<b>Data Type</b>	String
<b>Description</b>	This command sets a four-character alphanumeric (ASCII) code to be transmitted.
<b>Example</b>	See example for <code>ATS2.Bits.Pro.LocalAddress</code> .

---

**ATS2.Bits.Pro.Emphasis****Property**

<b>Syntax</b>	<code>ATS2.Bits.Pro.Emphasis</code>
<b>Result</b>	Constant  <i>apbBitsProEmpNotIndicated</i> Not Indicated  <i>apbBitsProEmpNone</i> None  <i>apbBitsProEmp5015</i> 50/15 uS  <i>apbBitsProEmpJ17</i> CCITT J.17
<b>Description</b>	This command sets the Emphasis parameter encoded in the Professional Status Bits.
<b>Example</b>	See example for <code>ATS2.Bits.Pro.LocalAddress</code> .

---

**ATS2.Bits.Pro.Flag****Property**

<b>Syntax</b>	<code>ATS2.Bits.Pro.Flag</code> (ByVal <i>FlagId</i> As Constant)
<b>Data Type</b>	Boolean  <i>True</i> Set <i>False</i> Clear

Parameter	Name	Description
	<i>FlagId</i>	<i>apbFlag0-5</i> = Select flag representing bits 0 - 5. <i>apbFlag6-13</i> = Select flag representing bits 6 - 13. <i>apbFlag14-17</i> = Select flag representing bits 14 - 17. <i>apbFlag18-21</i> = Select flag representing bits 18 - 21.
<b>Description</b>	This command sets or clears the Reliability Flag for the specified bytes.  This flag is to be set if useful information is not being transmitted in the corresponding status bytes.  Note that the Reliability Flags are not indications of the quality of the signal, but are simply a way for the transmitting device to tell the receiving device whether or not the information received in each group of six status bytes is valid.	
<b>Example</b>	See example for <code>ATS2.Bits.Pro.LocalAddress</code> .	

---

## ATS2.Bits.Pro.FreqMode

## Property

<b>Syntax</b>	<code>ATS2.Bits.Pro.FreqMode</code>
<b>Data Type</b>	Constant  <i>apbBitsProFreqUnLock</i> Unlocked  <i>apbBitsProFreqLock</i> Locked
<b>Description</b>	This command sets the Frequency Mode parameter encoded in the Professional Status Bits.
<b>Example</b>	See example for <code>ATS2.Bits.Pro.LocalAddress</code> .

---

## ATS2.Bits.Pro.LocalAddress

## Property

<b>Syntax</b>	<code>ATS2.Bits.Pro.LocalAddress</code>
<b>Data Type</b>	Long

**Description** This command sets the Local Address parameter encoded in the Professional Status Bits bytes 14-17.

The Local Address is a timer function defined in the Professional standard only.

**See Also** `ATS2.Bits.Pro.AddressAuto`

**Example**

```
Sub Main
  'other setup code ...
  ATS2.Bits.XmitChannel = apbBitsXmitChB
  ATS2.Bits.Mode = apbBitsModePro
  ATS2.Bits.Pro.AudioMode = apbBitsProNonAudio
  ATS2.Bits.Pro.Emphasis = apbBitsProEmp5015
  ATS2.Bits.Pro.FreqMode = apbBitsProFreqUnLock
  ATS2.Bits.Pro.SampleFreq = apbBitsPro_48k
  ATS2.Bits.Pro.ChMode = apbBitsProStereo
  ATS2.Bits.Pro.UserBits = apbBitsProUBitsUser
  ATS2.Bits.Pro.AuxBits = apbBitsPro24BitMain
  ATS2.Bits.Pro.WordLength = apbBitsProLen24
  ATS2.Bits.Pro.RefSignal = apbBitsProGrade2
  ATS2.Bits.Pro.Origin = "ATS2"
  ATS2.Bits.Pro.Destination = "TEST"
  ATS2.Bits.Pro.LocalAddressAuto = False
  ATS2.Bits.Pro.LocalAddress = 123456
  ATS2.Bits.Pro.TimeOfDay = 1234
  ATS2.Bits.Pro.Flag(apbFlag0_5) = True
  ATS2.Bits.Pro.Flag(apbFlag6_13) = True
  ATS2.Bits.Pro.Flag(apbFlag14_17) = False
  ATS2.Bits.Pro.Flag(apbFlag18_21) = True
  ATS2.Bits.Pro.CrcEnable = True
  'Rest of program
End Sub
```

---

## ATS2.Bits.Pro.LocalAddressAuto

## Property

**Syntax** `ATS2.Bits.Pro.LocalAddressAuto`

**Data Type** Boolean

*True*                      Enabled

*False*                      Disabled

- Description**      This command enables or disables automatic selection of the Local Address and Time Of Day values.
- If the Local Address Auto box via this command is enabled, both the Local Address value transmitted (bytes 14-17) and the Time of Day value (bytes 18-21) are the count, in samples, of the elapsed time since the Professional format of status bytes was selected or the Auto box was checked (whichever was later). If the Auto box is not checked, an entry field is displayed to the right of the Auto box. A number may be entered into this field via the `ATS2.Bits.Pro.LocalAddress` command and the number will be continuously transmitted as the Local Address code in the status bytes.
- See Also**            `ATS2.Bits.Pro.LocalAddress`,  
`ATS2.Bits.Pro.TimeOfDay`
- Example**            See example for `ATS2.Bits.Pro.LocalAddress`.

---

## ATS2.Bits.Pro.Origin

**Property**

- Syntax**            `ATS2.Bits.Pro.Origin`
- Data Type**        String
- Description**      This command sets a four-character alphanumeric (ASCII) code to be transmitted.
- Example**            See example for `ATS2.Bits.Pro.LocalAddress`.

---

## ATS2.Bits.Pro.RefSignal

**Property**

- Syntax**            `ATS2.Bits.Pro.RefSignal`
- Data Type**        Constant
- apbBitsProNotSignal*  
Nor a ref. Signal

*apbBitsProGrade1*  
Grade 1

*apbBitsProGrade2*  
Grade 2

*apbBitsProReserved*  
Reserved

**Description** This command sets the ReferenceSignal parameter encoded in the Professional Status Bits.

**Example** See example for `ATS2.Bits.Pro.LocalAddress`.

---

## ATS2.Bits.Pro.SampleFreq

## Property

**Syntax** `ATS2.Bits.Pro.SampleFreq`

**Data Type** Constant

*apbBitsPro\_NotIndicated*  
Not Indicated

*apbBitsPro\_48k*  
48 kHz

*apbBitsPro\_44k*  
44.1 kHz

*apbBitsPro\_32k*  
32 kHz

*apbBitsPro\_192k*  
192 kHz

*apbBitsPro\_192k1001*  
192/1.001 kHz

*apbBitsPro\_176k*  
176.4 kHz

*apbBitsPro\_176k1001*  
176.4/1.001 kHz



```

apbBitsPro_96k
    96 kHz
apbBitsPro_96k1001
    96/1.001 kHz
apbBitsPro_88k
    88.2 kHz
apbBitsPro_88k1001
    88.2/1.001 kHz
apbBitsPro_48k1001
    48/1.001 kHz
apbBitsPro_44k1001
    44.1/1.001 kHz
apbBitsPro_32k1001
    32/1.001 kHz
apbBitsPro_24k
    24 kHz
apbBitsPro_24k1001
    24/1.001 kHz
apbBitsPro_22k
    22.05 kHz
apbBitsPro_22k1001
    22.05/1.001 kHz

```

**Description** This command sets the Frequency parameter encoded in the Professional Status Bits.

**Example** See example for `ATS2.Bits.Pro.LocalAddress`.

---

## ATS2.Bits.Pro.TimeOfDay

## Property

**Syntax** `ATS2.Bits.Pro.TimeOfDay`

**Data Type** Long

<b>Description</b>	This command sets the Time Of Day parameter encoded in the Professional Status Bits bytes 18-21.
<b>See Also</b>	<code>ATS2.Bits.Pro.AddressAuto</code>
<b>Example</b>	See example for <code>ATS2.Bits.Pro.LocalAddress</code> .

---

## ATS2.Bits.Pro.UserBits

## Property

**Syntax** `ATS2.Bits.Pro.UserBits`

**Data Type** Constant

*apbBitsProUBitsNone*  
None

*apbBitsProUBits192*  
192-bit block

*apbBitsProUBitsReserved*  
Reserved

*apbBitsProUBitsUser*  
User defined

**Description** This command sets the User Bits parameter encoded in the Professional Status Bits.

**Example** See example for `ATS2.Bits.Pro.LocalAddress`.

---

## ATS2.Bits.Pro.WordLength

## Property

**Syntax** `ATS2.Bits.Pro.WordLength`

**Data Type** Constant

The following list contains the selections relevant to the `ATS2.Bits.Pro.AuxBits` command "20-bit not defined" selection.

*apbBitsProLenNotIndicated*  
Not Indicated

<i>apbBitsProLen20</i>	20 bits
<i>apbBitsProLen19</i>	19 bits
<i>apbBitsProLen18</i>	18 bits
<i>apbBitsProLen17</i>	17 bits
<i>apbBitsProLen16</i>	16 bits

The following list contains the selections relevant to the `ATS2.Bits.Pro.AuxBits` command "24-bit main audio" selection.

<i>apbBitsProLenNotIndicated</i>	Not Indicated
<i>apbBitsProLen24</i>	24 bits
<i>apbBitsProLen23</i>	23 bits
<i>apbBitsProLen22</i>	22 bits
<i>apbBitsProLen21</i>	21 bits
<i>apbBitsProLen20</i>	20 bits

**Description** This command sets the Audio Word Length parameter encoded in the Professional Status Bits.

**Example** See example for `ATS2.Bits.Pro.LocalAddress`.

**ATS2.Bits.RefSignalRdg****Property**

**Syntax** `ATS2.Bits.RefSignalRdg (ByVal Channel As Constant, [Optional ByVal String As Variant])`

**Data Type** Integer

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

Result	Constant	Description
	0	Not a ref. Signal
	1	Grade 1
	2	Grade 2
	3	Reserved

**Description** This command returns the Status Bits Reference Signal setting from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```
Sub Main
    If ATS2.Bits.ModeRdg(apbChA) = apbBitsModePro Then
        Debug.Print "Reference Signal Reading = ";
        Select Case (ATS2.Bits.RefSignalRdg(apbChA))
            Case apbBitsProNotSignal
                Debug.Print "Not a ref. Signal"
            Case apbBitsProGrade1
                Debug.Print "Grade 1"
            Case apbBitsProGrade2
```

```

        Debug.Print "Grade 2"
    Case apbBitsProReserved
        Debug.Print "Reserved"
    End Select
End If
End Sub

```

**Output**                    Reference Signal Reading = Not a ref. Signal

## ATS2.Bits.SampleFreqRdg

## Property

**Syntax**                    **ATS2.Bits.SampleFreqRdg** (ByVal *Channel* As Constant, [Optional ByVal *String* As Variant])

**Data Type**                Integer

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

Result	Constant	Description
--------	----------	-------------

**The following list is for Consumer Mode:**

<i>apbBitsConFreq_48k</i>	48 kHz
<i>apbBitsConFreq_41000</i>	44.1 kHz
<i>apbBitsConFreq_32k</i>	32 kHz
<i>apbBitsConFreq_24k</i>	24 kHz
<i>apbBitsConFreq_22k</i>	22.05 kHz
<i>apbBitsConFreq_192k</i>	192 kHz

*apbBitsConFreq\_176k*

176.4 kHz

*apbBitsConFreq\_96k*

96 kHz

*apbBitsConFreq\_882k*

88.2 kHz

**The following list is for Professional Mode:**

*apbBitsPro\_NotIndicated*

Professional Not Indicated

*apbBitsPro\_48k*

48 kHz

*apbBitsPro\_44k*

44.1 kHz

*apbBitsPro\_32k*

32 kHz

*apbBitsPro\_192k*

192 kHz

*apbBitsPro\_192k1001*

192/1.001 kHz

*apbBitsPro\_176K*

176.4 kHz

*apbBitsPro\_176k1001*

176.4/1.001 kHz

*apbBitsPro\_96k*

96 kHz

*apbBitsPro\_96k1001*

96/1.001 kHz

*apbBitsPro\_88k*

88.2 kHz

*apbBitsPro\_88k1001*

88.2/1.001 kHz

```

apbBitsPro_48k1001
    48/1.001 kHz
apbBitsPro_44k1001
    44.1/1.001 kHz
apbBitsPro_32k1001
    32/1.001 kHz
apbBitsPro_24k
    24 kHz
apbBitsPro_24k1001
    24/1.001 kHz
apbBitsPro_22k
    22.05 kHz
apbBitsPro_22k1001
    22.05/1.001 kHz

```

**Description** This command returns the Status Bits Sample Frequency from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```

Sub Main
    Debug.Print "Sample Freq Reading = ";
    Select Case (ATS2.Bits.SampleFreqRdg(apbChA))
        Case apbBitsCon_48k
            Debug.Print "Consumer 48 kHz"
        Case apbBitsCon_44100
            Debug.Print "Consumer 44.1 kHz"
        Case apbBitsCon_32k
            Debug.Print "Consumer 32 kHz"
        Case apbBitsPro_NotIndicated
            Debug.Print "Professional Not Indicated"
        Case apbBitsPro_96k1001
            Debug.Print "Professional 96/1.001 kHz"
        Case apbBitsPro_96k
            Debug.Print "Professional 96 kHz"
    End Select

```

```

Case apbBitsPro_88k1001
    Debug.Print "Professional 88.2/1.001 kHz"
Case apbBitsPro_88k
    Debug.Print "Professional 88.2 kHz"
Case apbBitsPro_48k1001
    Debug.Print "Professional 48/1.001 kHz"
Case apbBitsPro_48k
    Debug.Print "Professional 48 kHz"
Case apbBitsPro_44k1001
    Debug.Print "Professional 44.1/1.001 kHz"
Case apbBitsPro_44k
    Debug.Print "Professional 44.1 kHz"
Case apbBitsPro_32k1001
    Debug.Print "Professional 32/1.001 kHz"
Case apbBitsPro_32k
    Debug.Print "Professional 32 kHz"
Case apbBitsPro_24k1001
    Debug.Print "Professional 24/1.001 kHz"
Case apbBitsPro_24k
    Debug.Print "Professional 24 kHz"
Case apbBitsPro_22k1001
    Debug.Print "Professional 22.05/1.001 kHz"
Case apbBitsPro_22k
    Debug.Print "Professional 22.05 kHz"
Case apbBitsPro_192k1001
    Debug.Print "Professional 192/1.001 kHz"
Case apbBitsPro_192k
    Debug.Print "Professional 192 kHz"
Case apbBitsPro_176k1001
    Debug.Print "Professional 176.4/1.001 kHz"
Case apbBitsPro_176k
    Debug.Print "Professional 176.4 kHz"
End Select
End Sub

```

---

## ATS2.Bits.SourceNumRdg

## Property

**Syntax**      **ATS2.Bits.SourceNumRdg** (ByVal *Channel* As Constant, [Optional ByVal *String* As Variant])



**Data Type** Integer

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

<b>Result</b>	<b>Constant</b>	<b>Description</b>
	<i>apbBitsCon_DontCare</i>	Don't Care
	<i>apbBitsCon_1</i>	Source number 1
	<i>apbBitsCon_2</i>	Source number 2
	<i>apbBitsCon_3</i>	Source number 3
	<i>apbBitsCon_4</i>	Source number 4
	<i>apbBitsCon_5</i>	Source number 5
	<i>apbBitsCon_6</i>	Source number 6
	<i>apbBitsCon_7</i>	Source number 7
	<i>apbBitsCon_8</i>	Source number 8
	<i>apbBitsCon_9</i>	Source number 9
	<i>apbBitsCon_10</i>	Source number 10
	<i>apbBitsCon_11</i>	Source number 11

*apbBitsCon\_12*  
Source number 12

*apbBitsCon\_13*  
Source number 13

*apbBitsCon\_14*  
Source number 14

*apbBitsCon\_15*  
Source number 15

**Description** This command returns the Status Bits Source Number from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```
Sub Main
  If ATS2.Bits.ModeRdg(apbChA) = apbBitsModeCons Then
    Debug.Print "Source Number Reading = ";
    Select Case (ATS2.Bits.SourceNumRdg(apbChA))
      Case apbBitsCon_DontCare
        Debug.Print "Don't care"
      Case apbBitsCon_1
        Debug.Print "1"
      Case apbBitsCon_2
        Debug.Print "2"
      Case apbBitsCon_3
        Debug.Print "3"
      Case apbBitsCon_4
        Debug.Print "4"
      Case apbBitsCon_5
        Debug.Print "5"
      Case apbBitsCon_6
        Debug.Print "6"
      Case apbBitsCon_7
        Debug.Print "7"
      Case apbBitsCon_8
        Debug.Print "8"
      Case apbBitsCon_9
```

```

        Debug.Print "9"
    Case apbBitsCon_10
        Debug.Print "10"
    Case apbBitsCon_11
        Debug.Print "11"
    Case apbBitsCon_12
        Debug.Print "12"
    Case apbBitsCon_13
        Debug.Print "13"
    Case apbBitsCon_14
        Debug.Print "14"
    Case apbBitsCon_15
        Debug.Print "15"
End Select
End If
End Sub

```

## ATS2.Bits.StatusXferToString

## Method

**Syntax**      **ATS2.Bits.StatusXferToString** (ByVal *Channel* As Constant)

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

**Result**      String

**Description**      This command transfers the contents of the Status Bits into an string. This enable the programmer to extract all of the status information from an single measurement.

**See Also**      ATS2.Bits.XmitData

**Example**

```

'#uses"ConstantStrings.at sb"
Sub Main
    With ATS2.Bits
        Channel_A_Status = .StatusXferToString(apbChA)
    End With
End Sub

```

```

Mode = .ModeRdg(apbChA, Channel_A_Status)
If Mode = apbBitsModeCons Then
Debug.Print "Mode = Consumer"
    Debug.Print "Audio Mode = " & _
        (.AudioModeRdg(apbChA, Channel_A_Status))
    Debug.Print "Copyright = " & _
        (.CopyrightRdg(apbChA, Channel_A_Status))
    Debug.Print "Emphasis = " & _
        (.EmphRdg(apbChA, Channel_A_Status))
    Debug.Print "Channel Mode = " & _
        (.ChModeRdg(apbChA, Channel_A_Status))
    Debug.Print "Category Code = " & _
        (.CategoryRdg(apbChA, Channel_A_Status))
    Debug.Print "Source Number = " & _
        (.SourceNumRdg(apbChA, Channel_A_Status))
    Debug.Print "Channel Number = " & _
        (.ChNumRdg(apbChA, Channel_A_Status))
    Debug.Print "Sample Frequency = " & _
        (.SampleFreqRdg(apbChA, Channel_A_Status))
    Debug.Print "Clock Accuracy = " & _
        (.ClockAccuracyRdg(apbChA,
Channel_A_Status))

ElseIf Mode = apbBitsModePro then
Debug.Print "Mode = Professional"
    Debug.Print "Audio Mode = " & TextStr _
        (.AudioModeRdg(apbChA, Channel_A_Status))
    Debug.Print "Emphasis = " & _
        (.EmphRdg(apbChA, Channel_A_Status))
    Debug.Print "Frequency Mode = " & _
        (.FreqModeRdg(apbChA, Channel_A_Status))
    Debug.Print "Sample Frequency = " & _
        (.SampleFreqRdg(apbChA, Channel_A_Status))
    Debug.Print "Channel Mode = " & _
        (.ChModeRdg(apbChA, Channel_A_Status))
    Debug.Print "User Bits = " & _
        (.UserBitsRdg(apbChA, Channel_A_Status))
    Debug.Print "Aux Bits = " & _
        (.AuxBitsRdg(apbChA, Channel_A_Status))
    Debug.Print "Word Length = " & _
        (.WordLengthRdg(apbChA, Channel_A_Status))

```

```

Debug.Print "Ref Signal = " & _
    (.RefSignalRdg(apbChA, Channel_A_Status))
Debug.Print "Origin Code = " & _
    .OriginRdg(apbChA, Channel_A_Status)
Debug.Print "Destination Code = " & _
    .DestinationRdg(apbChA, Channel_A_Status)
Debug.Print "Local Address = " & _
    .LocalAddressRdg(apbChA, Channel_A_Status)
Debug.Print "Time Of Day = " & _
    .TimeOfDayRdg(apbChA, Channel_A_Status)
Debug.Print "Flag 0-5 = " & _
    .FlagRdg(apbChA, apbFlag0_5, _
    Channel_A_Status)
Debug.Print "Flag 6-13 = " & _
    .FlagRdg(apbChA, apbFlag6_13, _
    Channel_A_Status)
Debug.Print "Flag 14-17 = " & _
    .FlagRdg(apbChA, apbFlag14_17, _
    Channel_A_Status)
Debug.Print "Flag 18-21 = " & _
    .FlagRdg(apbChA, apbFlag18_21, _
    Channel_A_Status)
Debug.Print "Crc Valid = " & _
    .CrcRdg(apbChA, Channel_A_Status)
    End If
    End With
End Sub

```

## ATS2.Bits.TimeOfDayRdg

## Property

**Syntax**      **ATS2.Bits.TimeOfDayRdg** (ByVal *Channel* As Constant, [Optional ByVal *String* As Variant])

**Data Type**      Long

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

**Description** This command returns the Status Bits Time Of Day code from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.Bits.Mode = apbBitsModePro
    ATS2.Bits.Pro.LocalAddressAuto = False
    ATS2.Bits.Pro.TimeOfDay = 123456789
    ATS.S2Dio.InFormat = apbDioInGenMon
    Wait .5
    Debug.Print "Ch A Time Of Day Reading = " & _
        ATS2.Bits.TimeOfDayRdg(apbChA)
    Debug.Print "Ch B Time Of Day Reading = " & _
        ATS2.Bits.TimeOfDayRdg(apbChB)
End Sub
```

**Output**

```
Ch A Time Of Day Reading = 123456789
Ch B Time Of Day Reading = 123456789
```

---

## ATS2.Bits.UserBitsRdg

## Property

**Syntax** `ATS2.Bits.UserBitsRdg` (ByVal *Channel* As Constant, [Optional ByVal *String* As Variant])

**Data Type** Integer

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

Result	Constant	Description
	<i>apbBitsProUBitsNone</i>	None
	<i>apbBitsProUBits192</i>	192-bit block
	<i>apbBitsProUBitsReserved</i>	Reserved
	<i>apbBitsProUBitsUser</i>	User defined

**Description** This command returns the Status Bits User Bits from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```

Sub Main
  If ATS2.Bits.ModeRdg(apbChA) = apbBitsModePro Then
    Debug.Print "User Bits Reading = ";
    Select Case (ATS2.Bits.UserBitsRdg(apbChA))
      Case apbBitsProUBitsNone
        Debug.Print "None"
      Case apbBitsProUBits192
        Debug.Print "192-bit block"
      Case apbBitsProUBitsReserved
        Debug.Print "Reserved"
      Case apbBitsProUBitsUser
        Debug.Print "User defined"
    End Select
  End If
End Sub

```

---

## ATS2.Bits.WordLengthRdg

## Property

**Syntax** `ATS2.Bits.WordLengthRdg (ByVal Channel As Constant, [Optional ByVal String As Variant])`

**Data Type** Integer

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B
	<i>String</i>	Optional string containing status bit information.

<b>Result</b>	<b>Constant</b>	<b>Description</b>
	<i>apbBitsProLenNotIndicated</i>	Not Indicated
	<i>apbBitsProLen24</i>	24 bits
	<i>apbBitsProLen23</i>	23 bits
	<i>apbBitsProLen22</i>	22 bits
	<i>apbBitsProLen20</i>	20 bits
	<i>apbBitsProLen19</i>	19 bits
	<i>apbBitsProLen18</i>	18 bits
	<i>apbBitsProLen17</i>	17 bits
	<i>apbBitsProLen16</i>	16 bits

**Description** This command returns the Status Bits Word Length from an optional string or from the AES/EBU data stream. When the optional string parameter is included the command uses the designated string as the source for the reading. The string is obtained by using the `ATS2.Bits.StatusXferToString` command.

**See Also** `ATS2.Bits.StatusXferToString`

**Example** Sub Main



```

If ATS2.Bits.ModeRdg(apbChA) = apbBitsModePro Then
  Debug.Print "Word Length Reading = ";
  Select Case (ATS2.Bits.WordLengthRdg(apbChA))
    Case apbBitsProLenNotIndicated
      Debug.Print "Not Indicated"
    Case apbBitsProLen24
      Debug.Print "24 bits"
    Case apbBitsProLen23
      Debug.Print "23 bits"
    Case apbBitsProLen22
      Debug.Print "22 bits"
    Case apbBitsProLen21
      Debug.Print "21 bits"
    Case apbBitsProLen20
      Debug.Print "20 bits"
    Case apbBitsProLen19
      Debug.Print "19 bits"
    Case apbBitsProLen18
      Debug.Print "18 bits"
    Case apbBitsProLen17
      Debug.Print "17 bits"
    Case apbBitsProLen16
      Debug.Print "16 bits"
  End Select
End If
End Sub

```

---

## ATS2.Bits.XmitChannel

## Property

**Syntax**            **ATS2.Bits.XmitChannel**

**Data Type**        Constant

*apbBitsXmitChA*

A

*apbBitsXmitChB*

B

*apbBitsXmitChAB*

A & B

**Description** This command sets the Transmit Channel.

**Example** See example for `ATS2.Bits.Pro.LocalAddress`.

---

## ATS2.Bits.XmitStatus

## Property

**Syntax** `ATS2.Bits.XmitStatus (ByVal Channel As Constant)`

**Data Type** String String containing status bit information.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Channel A apbChB = Channel B

**Description** This command transmits the status bits data contained in the string for the specified channel.

**See Also** `ATS2.Bits.StatusXferToString`

**Example**

```
Sub Main
    With ATS2.Bits
        Channel_A_Status = .StatusXferToString(apbChA)
        Channel_B_Status = .StatusXferToString(apbChA)

        'Your code goes here

        .XmitStatus(apbChA) = Channel_A_Status
        .XmitStatus(apbChB) = Channel_B_Status
    End With
End Sub
```

---

### ATS2.DCX.DcLevel

### Property

**Syntax**      **ATS2.DCX.DcLevel1** (ByVal *Output* As Constant, ByVal *Unit* As String)

**Data Type**      Double                      -10.5 to 10.5 Volts

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Output</i>	apbOutput1 = Select output 1 apbOutput2 = Select output 2
	<i>Unit</i>	The following units are available Vdc.

**Description**      This command sets the voltage at the DCX's selected DC output.

**Example**

```
Sub Main
    ATS.Application.PanelOpen(apbDCX, apbLarge)
    ATS2.DCX.DcOutput(apbOutput1) = True
    ATS2.DCX.DcLevel(apbOutput1, "Vdc") = 1.5
End Sub
```

**Comment**      This macro turns on the DCX's DC output number 1 and sets the level to 1.5 volts.

---

### ATS2.DCX.DcOutput

### Property

**Syntax**      **ATS2.DCX.DcOutput** (ByVal *Output* As Constant)

**Data Type**      Boolean

<i>True</i>	Connects the output to the front panel.
<i>False</i>	Disconnects the output from the front panel.

Parameter	Name	Description
	<i>Output</i>	apbOutput1 = Output 1 apbOutput2 = Output 2
<b>Description</b>	This command sets DC Volts output to On or Off.	
<b>Example</b>	See example for <code>ATS2.DCX.DcLevel</code> .	

## ATS2.DCX.DigInFormat

## Property

**Syntax**      `ATS2.DCX.DigInFormat`

**Data Type**    Constant

*apbDcx2sComp*  
2's Complement

*apbDcxBCD*  
BCD

**Description**    This command sets the format of the digital input.

The digital ports are 21 bits plus a sign bit.

The normal format is two's complement. This format combines the bits into a 22 bit word that follows normal two's complement conventions (-1 is represented as 3FFFFFF hex).

The BCD (Binary coded decimal) format is a signed magnitude representation ( -1 is represented as 200001 hex, -10 is 200010 hex, etc.). As is normal in the BCD format, each decimal digit is represented by 4 bits.

### Example

```
Sub Main
    ATS.Application.PanelOpen(apbDCX, apbLarge)
    ATS2.DCX.DigOutFormat = apbDcxBCD
    ATS2.DCX.DigInRdgRate = apbDcx_4
    ATS2.DCX.DigOut("dec") = 100
    ATS2.DCX.DigInFormat = apbDcxBCD
    Reading1 = ATS2.DCX.DigInRdg("dec")
    ATS2.DCX.DigOut("h(x)") = 100
    ATS2.DCX.DigOutScale = 2
```

```

    ATS2.DCX.DigInSettling(.20, .1, "Dec", 4, .05,
apbNone)
ATS2.DCX.DigInTrig
Do
Loop Until ATS2.DCX.DigInReady
Reading2 = ATS2.DCX.DigInRdg("dec")
ATS2.DCX.DigInScale = .5
Reading3 = ATS2.DCX.DigInRdg("g(x)")
NewLine$ = Chr(13)
a$= "Reading1 "+Left(Str$(Reading1),6)+"dec"
b$= "Reading2 "+Left(Str$(Reading2),6)+"dec"
c$= "Reading3 "+Left(Str$(Reading3),6)+"dec"
ATS.Prompt.Text = a$ & vbCR & b$ & vbCR $ & c$
ATS.Prompt.ShowWithContinue
Beep
Stop
End Sub

```

---

## ATS2.DCX.DigInRdg

## Property

**Syntax**            **ATS2.DCX.DigInRdg**(ByVal *Unit* As String)

**Data Type**        Double

Parameter	Name	Description
	<i>Unit</i>	The following units are available: dec, hex,oct, and g(x).

**Description**     This command returns a settled reading for the DCX-127 Digital In meter and zeros the ready count.

**See Also**         **ATS2.DCX.DigInReady**, **ATS2.DCX.DigInSettling**, **ATS2.DCX.DigInTrig**

**Example**          See example for **ATS2.DCX.DigInFormat**.

---

## ATS2.DCX.DigInRdgRate

## Property

**Syntax**            **ATS2.DCX.DigInRdgRate**

<b>Data Type</b>	Constant
	<i>apbDcx_External</i> External Strobe (Default).
	<i>apbDcx_4</i> 4 readings per second.
	<i>apbDcx_8</i> 8 readings per second.
	<i>apbDcx_16</i> 16 readings per second.
	<i>apbDcx_32</i> 32 readings per second.
<b>Description</b>	This command selects an internal or external strobe for the digital input.  If 0 is selected, the External Strobe available on pin (25) of the digital input connector is used to trigger each new reading. If 1-4 is selected, an internal strobe is used at the specified rate.
<b>Example</b>	See example for <code>ATS2.DCX.DigInFormat</code> .

---

## ATS2.DCX.DigInReady

## Property

<b>Syntax</b>	<code>ATS2.DCX.DigInReady</code>
<b>Data Type</b>	Integer
	0                      Reading not ready.
	>0                     Reading ready.
<b>Description</b>	This command returns the DCX-127 Digital In settled reading ready count.  Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the <code>ATS2.DCX.DigInRdg</code> or <code>ATS2.DCX.DigInTrig</code> commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.DCX.DigInRdg` command will be guaranteed to return quickly.

**See Also** `ATS2.DCX.DigInRdg`, `ATS2.DCX.DigInSettling`, `ATS2.DCX.DigInTrig`

**Example** See example for `ATS2.DCX.DigInFormat`.

## ATS2.DCX.DigInScale

Property

**Syntax** `ATS2.DCX.DigInScale`

**Data Type** Double

**Description** This command sets the DCX-127 Digital Input Scale factor.

When g(x) units are selected at the Digital In display, ATS software computes the displayed value from the relationship

$$\text{display} = \text{measurement} * \text{Scale (g)}$$

where measurement is the decimal value of the binary data in the selected format and Scale (g) is the value entered in the Scale (g) field just below the Digital In display.

**Example** See example for `ATS2.DCX.DigInFormat`.

## ATS2.DCX.DigInSettling

Method

**Syntax** `ATS2.Inst.Analyzer.DigInSettling (ByVal Tolerance As Double, ByVal Floor As Double, ByVal FloorUnit As String, ByVal Points As Integer, ByVal Delay As Double, ByVal Algorithm As Constant)`

**Parameter** See Appendix A for Settling Algorithm and parameter name descriptions.

**Description** This command sets the settling parameters for the `ATS2.DCX.DigInRdg` command.

**See Also** `ATS2.DCX.DigInFormat`, `ATS2.DCX.DigInRdg`,  
`ATS2.DCX.DigInReady`, `ATS2.DCX.DigInTrig`

**Example** See example for `ATS2.DCX.DigInFormat`.

## ATS2.DCX.DigInTrig

**Method**

**Syntax** `ATS2.DCX.DigInTrig`

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.DCX.DigInRdg` command. The reading in progress is aborted.

**See Also** `ATS2.DCX.DigInRdg`, `ATS2.DCX.DigInReady`,  
`ATS2.DCX.DigInSettling`

**Example** See example for `ATS2.DCX.DigInFormat`.

## ATS2.DCX.DigOut

**Property**

**Syntax** `ATS2.DCX.DigOut (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<code>Unit</code>	The following units are available dec, hex, oct, h(x).

**Description** This command sets the value of the DCX's digital output.  
The output format is either two's complement or BCD as set by the `ATS2.DCX.DigOutFormat` command.

**Example** See example for `ATS2.DCX.DigInFormat`.

## ATS2.DCX.DigOutFormat

**Property**

**Syntax** `ATS2.DCX.DigOutFormat`

**Data Type** Constant



*apbDcx2sComp*  
2's Complement

*apbDcxBCD*  
BCD

- Description** This command sets the format of the digital output. The digital ports are 21 bits plus a sign bit. The normal format is two's complement. This format combines the bits into a 22 bit word that follows normal two's complement conventions (-1 is represented as 3FFFFFF hex). The BCD (Binary coded decimal) format is a signed magnitude representation ( -1 is represented as 200001 hex, -10 is 200010 hex, etc.). As is normal in the BCD format, each decimal digit is represented by 4 bits.
- Example** See example for `ATS2.DCX.DigInFormat`.

## ATS2.DCX.DigOutScale

## Property

- Syntax** `ATS2.DCX.DigOutScale`
- Data Type** Double
- Description** When h(x) units are selected at the Digital Output control field, ATS software computes the actual transmitted value from the relationship
- $$\text{output value} = \text{entry value} * \text{Scale (h)}$$
- where entry value is the decimal value entered into the Digital Out numeric field and Scale (h) is the value entered in the Scale (h) field just below the Digital Out control field.
- Example** See example for `ATS2.DCX.DigInFormat`.

## ATS2.DCX.DmmMode

## Property

- Syntax** `ATS2.DCX.DmmMode`

<b>Data Type</b>	Constant
	<i>apbDcxOff</i>
	Off
	This command disconnects the DMM from the front panel jacks.
	This allows the DMM to be wired to the circuit under test yet not be connected until needed. This is so that there is no possibility of the DMM input characteristics degrading the results of any other measurements being made ATS.
	<i>apbDcxDCV</i>
	DC Volts
	<i>apbDcxOhms</i>
	Ohms

**Description** This command sets the DMM measurement mode.

**Example**

```
Sub Main
    ATS.Application.PanelOpen(apbDCX, apbLarge)
    ATS2.DCX.DmmRange = 2.0
    ATS2.DCX.DmmMode = apbDcxDCV
    ATS2.DCX.DmmRdgRate = apbDcxRdgRate_6
    ATS2.DCX.DmmSettling(1, .20, "Vdc", 3, .03,
apbNone)
    ATS2.DCX.DmmTrig
    Do
    Loop Until ATS2.DCX.DmmReady
    Reading1 = ATS2.DCX.DmmRdg ("Vdc")
    a$= "DMM Reading " & Left(Str$(Reading1),6) & "Vdc"
    ATS.Prompt.Text = a$ + vbCR
    ATS.Prompt.ShowWithContinue
    Beep
    Stop
End Sub
```

**Comment** This macro sets the DMM to volts, selects 2 Volt range, sets the reading rate, sets settling, triggers a New reading, waits For the New reading, and stores it In a variable called Reading1.

---

## ATS2.DCX.DmmOffset

## Property

**Syntax**            `ATS2.DCX.DmmOffset`

**Data Type**        Double

**Description**     When f(V) (function of Volts) or f(O) (function of Ohms) units are selected for the DMM, ATS software computes the value to display from the formula

$$\text{display} = (\text{measurement} + \text{Offset}) * \text{Scale}$$

The measurement term is the value which would be displayed in Volts or Ohms units. The Offset and Scale values are the contents of the fields with those names, at the top right of the DCX panel.

**See Also**            `ATS2.DCX.DmmScale`

**Example**            See example for `ATS2.DCX.DmmMode`.

---

## ATS2.DCX.DmmRange

## Property

**Syntax**            `ATS2.DCX.DmmRange`

**Data Type**        Double

**Description**     This command sets the DMM's input range and returns the nominal full scale of range in use.

The ranges for Ohms mode are:

2M, 200k, 20k, 2k, 200 Ohms

The ranges for Volts mode are:

500, 200, 20, 2.0, 0.2 Volts

A common use of this command is in fixing the input range by obtaining the range and then using that value for this command.

**Example**            See example for `ATS2.DCX.DmmMode`.

**ATS2.DCX.DmmRangeAuto****Property****Syntax**      `ATS2.DCX.DmmRangeAuto`**Data Type**      Boolean

<i>True</i>	Auto range
<i>False</i>	Fixed range

**Description**      This command sets the DCX-127 DMM input to Auto range or fixed range. Care must be taken when using Fixed range that the input signal does not exceed the selected range.**See Also**      `ATS2.DCX.DmmRange`**Example**

```
Sub Main
    ATS.Application.PanelOpen(apbDCX, apbLarge)
    ATS2.DCX.DmmRangeAuto = True
    ATS2.DCX.DmmMode = apbDcxDCV
    ATS2.DCX.DmmRdgRate = apbDcxRdgRate_25
    ATS2.DCX.DmmScale = 2
    ATS2.DCX.DmmOffset = 1
    ATS2.DCX.DmmSettling(1, .20, "Vdc", 3, .03,
apbNone)
    ATS2.DCX.DmmTrig
    Do
    Loop Until ATS2.DCX.DmmReady
    Reading1 = ATS2.DCX.DmmRdg("f(v)")
    a$= "DMM Reading " & Left(Str$(Reading1),6) _
        & "f(V)"
    ATS.Prompt.Text = a$ + vbCR
    ATS.Prompt.ShowWithContinue
    Beep
    Stop
End Sub
```

**ATS2.DCX.DmmRdg****Property****Syntax**      `ATS2.DCX.DmmRdg (ByVal Unit As String)`**Data Type**      Variant

Parameter	Name	Description
	<i>Unit</i>	The following units are available VDC, V(f) for the <code>ATS2.DCX.DmmMode</code> command DCV mode and Ohms, and f(O) for the Ohms mode.
<b>Description</b>		This command returns a settled reading for the DCX-127 Digital Multi meter(DMM) meter and zeros the ready count.
<b>See Also</b>		<code>ATS2.DCX.DmmMode</code> , <code>ATS2.DCX.DmmReady</code> , <code>ATS2.DCX.DmmSettling</code> , <code>ATS2.DCX.DmmTrig</code>
<b>Example</b>		See example for <code>ATS2.DCX.DmmMode</code> .

## ATS2.DCX.DmmRdgRate

## Property

<b>Syntax</b>	<code>ATS2.DCX.DmmRdgRate</code>
<b>Data Type</b>	Constant
	<code>apbDcxRdgRate_6</code> 6 readings per second.
	<code>apbDcxRdgRate_25</code> 25 readings per second.
<b>Description</b>	This command sets the DMM reading rate.
<b>Example</b>	See example for <code>ATS2.DCX.DmmMode</code> .

## ATS2.DCX.DmmReady

## Property

<b>Syntax</b>	<code>ATS2.DCX.DmmReady</code>
<b>Data Type</b>	Integer
	<code>0</code> Reading not ready.
	<code>&gt;0</code> Reading ready.
<b>Description</b>	This command returns the DCX-127 DMM settled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the `ATS2.DCX.DmmRdg` or `ATS2.DCX.DmmTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.DCX.DmmRdg` command will be guaranteed to return quickly.

**See Also** `ATS2.DCX.DmmRdg`, `ATS2.DCX.DmmSettling`,  
`ATS2.DCX.DmmTrig`

**Example** See example for `ATS2.DCX.DmmMode`.

---

## ATS2.DCX.DmmScale

**Property**

**Syntax** `ATS2.DCX.DmmScale`

**Data Type** Double

**Description** When f(V) (function of Volts) or f(O) (function of Ohms) units are selected for the DMM, ATS software computes the value to display from the following formula:

$$\text{display} = (\text{measurement} + \text{Offset}) * \text{Scale}$$

The measurement term is the value which would be displayed in Volts or Ohms units. The Offset and Scale values are the contents of the fields with those names, at the top right of the DCX panel.

**See Also** `ATS2.DCX.DmmOffset`

**Example** See example for `ATS2.DCX.DmmRangeAuto`.

---

## ATS2.DCX.DmmSettling

**Method**

**Syntax** `ATS2.DCX.DmmSettling` (*ByVal Tolerance As Double, ByVal Floor As Double, ByVal FloorUnit As String, ByVal Points As Integer, ByVal Delay As Double, ByVal Algorithm As Constant*)

<b>Parameter</b>	See Appendix A for Settling Algorithm and parameter name descriptions.
<b>Description</b>	This command sets the settling parameters for the <code>ATS2.DCX.DmmRdg</code> command.
<b>See Also</b>	<code>ATS2.DCX.DmmRdg</code> , <code>ATS2.DCX.DmmReady</code> , <code>ATS2.DCX.DmmTrig</code>
<b>Example</b>	See example for <code>ATS2.DCX.DmmMode</code> .

---

## ATS2.DCX.DmmTrig

Method

<b>Syntax</b>	<code>ATS2.DCX.DmmTrig</code>
<b>Description</b>	Causes a restart of the reading cycle and zeros the ready count for the <code>ATS2.DCX.DmmRdg</code> command. The reading in progress is aborted.
<b>See Also</b>	<code>ATS2.DCX.DmmRdg</code> , <code>ATS2.DCX.DmmReady</code> , <code>ATS2.DCX.DmmSettling</code>
<b>Example</b>	See example for <code>ATS2.DCX.DmmMode</code> .

---

## ATS2.DCX.GateDelay

Property

<b>Syntax</b>	<code>ATS2.DCX.GateDelay</code>
<b>Data Type</b>	Double      Valid settings are from 0.05 to 12.75 sec.
<b>Description</b>	This command sets the delay time for the delayed sweep gate, pin #1 on the DCX-127 Program Control Output port transitions low after the defined delay.  Note: When using long delays the sweep duration must be longer than the programmed delay for pin #1 to respond.
<b>Example</b>	<pre>Sub Main     ATS.Application.NewTest     ATS.Application.PanelOpen(apbDCX, apbLarge)</pre>

```

ATS2.DCX.GateDelay = 0.1
ATS2.AGen.OutputOn = True
ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
ATS.Sweep.Start
End Sub

```

---

## ATS2.DCX.PortOutput

## Property

**Syntax** **ATS2.DCX.PortOutput** (ByVal *Port* As Constant, ByVal *Unit* As String)

**Data Type** Long The number can be from 0 to 255. Larger numbers are truncated to 8 bits. This value can only be expressed as a decimal value.

Parameter	Name	Description
	<i>Port</i>	apbDcxA = Port A apbDcxB = Port B apbDcxC = Port C apbDcxD = Port D (J141)
	<i>Unit</i>	The following units are available: dec, hex, oct.

**Description** This command sets DCX-127 Ports A through D 8-bit output value.

**Example**

```

Sub Main
    ATS.Application.PanelOpen(apbDCX, apbLarge)
    ATS2.DCX.PortOutput(apbDcxA, "Dec") = 17
    ATS2.DCX.PortOutput(apbDcxB, "Hex") = 34
    ATS2.DCX.PortOutput(apbDcxC, "Oct") = 68
    ATS2.DCX.PortOutput(apbDcxD, "Oct") = 68
End Sub

```



# Chapter 23

## Digital Generator

### ATS2.DGen.Ampl

Property

**Syntax** `ATS2.DGen.Ampl (ByVal Channel As Constant, ByVal Unit As String)`

**Data Type** Double Valid amplitude settings are 0.0 to 100 %FS.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, PPM, Bits, Vrms, Vp, Vpp, dBu, dBV, dBr

**Description** This command sets the Digital Generator specified channel amplitude.

**See Also** `ATS2.DGen.Ampl`, `ATS2.DGen.RefVFS`

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS.Application.PanelClose(apbAnalogGen)
    ATS.Application.PanelOpen(apbDigitalGen, apbLarge)
    ATS2.DGen.Wfm.Sine.Stereo
    With ATS2.DGen
        .RefVFS("V") = 2
        .RefFreq("Hz") = 10e3
        .RefdBr("FFS") = 0.5
        .ChBTrackA = False
        .Freq(apbChA, "%Hz") = 50
        .Freq(apbChB, "%Hz") = 75
        .Ampl(apbChA, "dBr") = 0.5
        .Ampl(apbChB, "dBr") = 0.5
        .Invert(apbChA) = False
    End With
End Sub
```

```

        .Invert(apbChB) = False
        .Output(apbChA) = True
        .Output(apbChB) = True
        .DitherType = apbDGenTriangular
        .OutputOn = True
    End With

    ATS2.Dio.InFormat = apbDioInGenMon
    With ATS2.Inst.Analyzer.InputFormat = _
        apbInstInDigital
        Do While (.LevelReady(apbChA) = False) Or _
            (.LevelReady(apbChB) = False) Or _
            (.FreqReady(apbChA) = False) Or _
            (.FreqReady(apbChB) = False)
        Loop
        msg = "Ch A Level = " & Format _
            (.LevelRdg(apbChA, "V"), "#.00") & Chr(13)
        msg = msg & "Ch B Level = " & Format _
            (.LevelRdg(apbChB, "V"), "#.00") & Chr(13)
        msg = msg & "Ch A Freq = " & Format _
            (.FreqRdg(apbChA, "Hz"), "#.00") & Chr(13)
        msg = msg & "Ch B Freq = " & Format _
            (.FreqRdg(apbChB, "Hz"), "#.00")
    End With
    ATS.Prompt.Text = msg
    ATS.Prompt.ShowWithContinue
    Stop
End Sub

```

## ATS2.DGen.BurstInterval

## Property

**Syntax**      **ATS2.DGen.BurstInterval** (ByVal *Unit* As String)

**Data Type**      Double      2 - 65536 cycles or equivalent period based on sine waveform frequency.

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Cycles, sec

**Description** This command sets the number of cycles between the start of a burst and the start of the following burst. This number may be from 2 to 65535 cycles and must be greater than the number of ON cycles. If the number of cycles attempted is not greater than the ON cycles, the interval is not changed.

Note that the interval will occur immediately when this command is called if the burst is running.

**See Also** `ATS2.DGen.Wfm`, `ATS2.DGen.BurstLevel`,  
`ATS2.DGen.BurstOnTime`

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS.Application.PanelClose(apbAnalogGen)
    ATS.Application.PanelOpen(apbDigitalGen, apbLarge)
    ATS2.DGen.Wfm.Sine.Burst
    ATS2.DGen.Output = True
    ATS2.DGen.BurstInterval("Cycles") = 10
    ATS2.DGen.BurstOnTime("Cycles") = 5
    ATS2.DGen.BurstLevel("dB") = -40
    Interval = ATS2.DGen.BurstInterval("Cycles")
    Ontime = ATS2.DGen.BurstOnTime("Cycles")
    Level = ATS2.DGen.BurstLevel("%")
    Debug.Print "Burst Interval = " & Interval & _
        " cycles."
    Debug.Print "Burst ON time = " & Ontime & " _
        cycles."
    Debug.Print "Burst OFF time low level = " & _
        Format(Level, "0.0") & " %."
End Sub
```

**Output**

```
Burst Interval = 10 cycles.
Burst ON time = 5 cycles.
Burst OFF time low level = 1.0 %.
```

---

## ATS2.DGen.BurstLevel

**Property**

**Syntax** `ATS2.DGen.BurstLevel (ByVal Unit As String)`

<b>Data Type</b>	Double	Level of signal during burst off time. (0 - -80.25dB)
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	The following units are available X/Y, dB, %, PPM.
<b>Description</b>	This command sets the amplitude of the Digital Generator during the burst 'off' time. This is as a percentage of the 'on' amplitude and may range from 100.0 percent to .009716280 percent (-80.25 dB).	
<b>See Also</b>	ATS2.DGen.Wfm, ATS2.DGen.BurstInterval, ATS2.DGen.BurstOnTime	
<b>Example</b>	See example for ATS2.DGen.BurstInterval.	

---

## ATS2.DGen.BurstOnTime

## Property

<b>Syntax</b>	<b>ATS2.DGen.BurstOnTime</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	From 1 to ATS2.AGen.BurstInterval - 1.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Cycles, sec
<b>Description</b>	This command sets the number of cycles for the Digital Generator Burst On Time. This number may be from 1 to 65534 cycles and must be less than the number of interval cycles. If the number of cycles attempted is not less than the interval cycles, the ON time is not changed.	
<b>See Also</b>	ATS2.DGen.Wfm, ATS2.DGen.BurstInterval, ATS2.DGen.BurstLevel	
<b>Example</b>	See example for ATS2.DGen.BurstInterval.	

---

## ATS2.DGen.ChBTrackA

## Property

<b>Syntax</b>	<b>ATS2.DGen.ChBTrackA</b>
<b>Data Type</b>	Boolean

*True* ON, channel B amplitude tracks channel A amplitude.  
*False* OFF, channel B amplitude independent of channel A.

**Description** This command sets the Digital Generator channel B amplitude to the same amplitude as set for channel A.

**See Also** `ATS2.DGen.Ampl`, `ATS2.DGen.Ampl`

**Example** See example for `ATS2.DGen.Ampl`.

---

## ATS2.DGen.DitherType

## Property

**Syntax** `ATS2.DGen.DitherType`

**Data Type** Constant

*apbDGenTriangular*

Triangular: probability function dither has no noise modulation effect but produces a slightly worse output signal to noise ratio since its maximum amplitude is one LSB. This is normally the preferred choice.

*apbDGenRectangular*

Rectangular: probability function dither provides the best signal to noise due to its one-half LSB amplitude, but suffers from modulation noise effects.

*apbDgenShaped*

Shaped: is triangular probability distribution noise with a rising 6 dB/octave slope. This places most of the dither power at higher frequencies where some falls out of band of most devices and where the human hearing system is less sensitive.

*apbDgenNone*

None:

**Description** This command sets the Digital Generator Dither Type.

Dither amplitude is automatically set corresponding to the LSB of the value selected in the Output Resolution field or by the `ATS2.Dio.OutResolution` command.

Dither is random noise of one-half LSB (rectangular) or one LSB (triangular) in amplitude, added to the digital output to improve linearity, reduce distortion, and extend the dynamic range downwards below the theoretical undithered value. The amplitude at which dither is added is determined by the value entered in the Output Resolution field or by the `ATS2.Dio.OutResolution` command.

**Example** See example for `ATS2.DGen.Ampl`.

---

## ATS2.DGen.DualAmplRatio

## Property

**Syntax** `ATS2.DGen.DualAmplRatio (ByVal Unit As String)`

**Data Type** Double Valid settings are 0.00001% to 100%

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: %, dB, PPM, X/Y

**Description** This command sets the Digital Generator Dual waveform amplitude ratio. The amplitude of Frequency 2 is set relative to the main frequency.

**See Also** `ATS2.DGen.Freq`

---

## ATS2.DGen.EqAmpl

## Property

**Syntax** `ATS2.DGen.EqAmpl (ByVal Channel As Constant, ByVal Unit As String)`

**Data Type** Double Valid amplitude settings are 0.0 to 100 %FS.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, PPM, Bits, Vrms, Vp, Vpp, dBu, dBV, dBr

**Description** This command sets the Digital Generator post Eq amplitude.

**See Also** `ATS2.DGen.EqCurve`

**Example**

```
Sub Main
  ATS.Application.NewTest
  ATS.Application.PanelClose (apbAnalyzer)
  ATS.Application.PanelClose (apbAnalogGen)
  ATS.Application.PanelOpen (apbDigitalGen, apbLarge)
  ATS2.DGen.Ampl (apbChA, "dBFS") = -1.0
  ATS2.DGen.Ampl (apbChB, "dBFS") = -1.0
  ATS2.DGen.EqCurve ("75us-de.atsq", 1)
  ATS2.DGen.Wfm.Sine.EQ
  ATS2.DGen.EqAmpl (apbChA, "dBFS") = -1.0
  ATS2.DGen.EqAmpl (apbChB, "dBFS") = -1.0
  ATS2.DGen.OutputOn = True
  ATS.Sweep.Data(1).Id = 6014
  ATS.Sweep.Data(1).Top("dBFS") = 0.0
  ATS.Sweep.Data(1).Bottom("dBFS") = -25.0
  ATS.Sweep.Source(1).Id = 5102
  ATS.Sweep.Stereo = True
  ATS.Sweep.Start
End Sub
```

## ATS2.DGen.EqCurve

## Method

**Syntax** `ATS2.DGen.EqCurve (ByVal FileName As String, ByVal Column As Integer)`

**Parameter**

Name	Description
<i>FileName</i>	Long Path and File Names permitted up to 128 characters. The file must be an ATS Eq file (.adq).
<i>Column</i>	0 = Source 1 settings. 1 = Data 1 measurements. 2 = Data 2 measurements. 3 = Data 3 measurements. 4 = Data 4 measurements. 5 = Data 5 measurements. 6 = Data 6 measurements.

7 = Source 2 settings.

<b>Result</b>	Boolean
	<i>True</i> File open successfull.
	<i>False</i> File open failed.
<b>Description</b>	This command attaches a Eq file to the Digital Generator. Values in the file will be used as multiply factors in czlcutating the Digital Generator Amplitude values.
<b>See Also</b>	ATS2.DGen.EqAmpl
<b>Example</b>	See example for ATS2.DGen.EqAmpl.

## ATS2.DGen.EqCurveColumn

### Get Only Property

**Syntax** `ATS2.DGen.EqCurveColumn (ByVal Data As Integer)`

**Data Type** Integer Column number.

Parameter	Name	Description
	<i>Data</i>	Number of the Sweep Data (1-6) of the data in memory.

**Description** This command returns the column number in the attached file used in the Digital Generator EqCurve waveform selection.

**See Also** `ATS.Sweep.Data.AutoDiv`, `ATS.Sweep.Data.LogLin`

## ATS2.DGen.EqCurveFilename

### Get Only Property

**Syntax** `ATS2.DGen.EqCurveFilename`

**Data Type** Integer Any valid DOS filename and extension.

**Description** This command returns the File Name of the attached file used for the Digital Generator EqCurve waveform selection.

**See Also** `ATS2.DGen.EqCurve`, `ATS2.DGen.EqCurveColumn`



**ATS2.DGen.Freq****Property**

<b>Syntax</b>	<b>ATS2.DGen.Freq</b> (ByVal <i>Channel</i> As Constant, ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	Valid frequency settings for the Hz unit and sine waveform are 10 - 22.5kHz for the 48kHz sample rate.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM
<b>Description</b>	This command sets the Digital Generator frequency for the Sine waveforms.	

**ATS2.DGen.IMFreq****Property**

<b>Syntax</b>	<b>ATS2.DGen.IMFreq</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	For a SMPTE mode waveform, this is the lower frequency tone. The following frequency range is available for SMPTE and CCIF IMD waveforms for the 48kHz sample rate: 10 Hz to 22.5kHz.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command. Hz
<b>Description</b>	This command sets the Digital Generator IMD frequency. The frequency passed is rounded to the closest available value.  Set the Digital Generator waveform to IMD (SMPTE/DIN) before calling this command.	
<b>See Also</b>	ATS2.DGen.Wfm, ATS2.DGen.IMCenterFreq	

**ATS2.DGen.IMHighFreq****Property**

**Syntax**      `ATS2.DGen.IMHighFreq (ByVal Unit As String)`

**Data Type**    Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command. Hz

**Description**    This command sets the Digital Generator IMD High Frequency. The frequency passed is rounded to the closest available value.

Set the Digital Generator waveform to an IMD SMPTE/DIN before calling this command.

**See Also**      `ATS2.DGen.Wfm`, `ATS2.DGen.IMFreq`

**ATS2.DGen.Invert****Property**

**Syntax**

`ATS2.DGen.Invert (ByVal Channel As Constant)`

Data Type	Boolean	
	<i>True</i>	Invert channel A output.
	<i>False</i>	Normal non-inverting output.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description**    This command sets the selected channel output to normal polarity or inverted polarity (180 degrees out of phase).

**Example**      See example for `ATS2.DGen.Ampl`.

## ATS2.DGen.Offset

## Property

**Syntax** `ATS2.DGen.Offset (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits, Vrms, Vp, Vpp, dBu, dBV, dBr, dBrlnv

**Description** This command sets the Digital Generator Sine + Offset waveform Offset value.

**See Also** `ATS2.DGen.Wfm`

## ATS2.DGen.Output

## Property

**Syntax** `ATS2.DGen.Output (ByVal Channel As Constant)`

**Data Type** Boolean

<i>True</i>	ON.
<i>False</i>	OFF.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** This command sets the Digital Generator Output to ON or OFF. The command returns a TRUE if the output is ON and FALSE if the output is OFF.

**See Also** `ATS2.DGen.OutputOn`

**Example** See example for `ATS2.DGen.Ampl`.

## ATS2.DGen.OutputOn

## Property

<b>Syntax</b>	<code>ATS2.DGen.OutputOn</code>
<b>Data Type</b>	Boolean
	<code>True</code> <code>On</code>
	<code>False</code> <code>Off</code>
<b>Description</b>	This command sets the Digital Generator channel A and B outputs to ON or OFF if they have been individually enabled by the <code>ATS2.DGen.Output</code> command.
<b>See Also</b>	<code>ATS2.DGen.Output</code>
<b>Example</b>	See example for <code>ATS2.DGen.Ampl</code> .

## ATS2.DGen.Phase

## Property

<b>Syntax</b>	<code>ATS2.DGen.Phase (ByVal Unit As String)</code>				
<b>Data Type</b>	Double				
<b>Parameter</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>Unit</code></td> <td>String that designates the desired unit. The following units are valid for this command: deg</td> </tr> </tbody> </table>	Name	Description	<code>Unit</code>	String that designates the desired unit. The following units are valid for this command: deg
Name	Description				
<code>Unit</code>	String that designates the desired unit. The following units are valid for this command: deg				
<b>Description</b>	<p>This command sets the Digital Generator Phase value.</p> <p>Set the Digital Generator waveform to Sine Var Phase before calling this command.</p>				
<b>See Also</b>	<code>ATS2.DGen.Wfm</code>				
<b>Example</b>	<pre>Sub Main     ATS.Application.NewTest     ATS.Application.PanelClose (apbAnalogGen)     ATS.Application.PanelOpen (apbDigitalGen, apbSmall)     ATS2.DGen.WfmSine.VarPhase     <b>ATS2.DGen.Phase</b> ("deg") = 90.0     ATS2.DGen.OutputOn = True     ATS2.Dio.InFormat = apbDioInGenMon     ATS2.Inst.Analyzer.InputFormat = apbInstInDigital</pre>				

```

ATS2.Inst.Analyzer.FuncMode = apbAnlrPhase
ATS.Sweep.Data(1).Id = 6365
ATS.Sweep.SinglePoint = True
ATS.Sweep.Start
Debug.Print "Channel B is " &
Format(ATS.Data.Value(0,1,0,"deg"),"##.000") & " deg
relative to channel A"
End Sub

```

**Output** Channel B is 90.001 deg relative to channel A.

## ATS2.DGen.RefdBr

## Property

<b>Syntax</b>	<b>ATS2.DGen.RefdBr</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	Amplitude value.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits, V, Vp, Vpp, dBu, dBV
<b>Description</b>	This command sets the zero dBr value for the Digital Generator dBr unit.	
<b>See Also</b>	ATS2.DGen.Ampl, ATS2.DGen.RefVFS	
<b>Example</b>	See example for ATS2.DGen.Ampl.	

## ATS2.DGen.RefFreq

## Property

<b>Syntax</b>	<b>ATS2.DGen.RefFreq</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command: Hz

**Description** This command sets the Digital Generator relative frequency reference value. This reference is used for all the Digital Generator relative frequency units (F/R, dHz, %Hz, cent, octs, decs, d%, dPPM)

**See Also** `ATS2.DGen.Freq`

**Example** See example for `ATS2.DGen.Ampl`.

## ATS2.DGen.RefVFS

## Property

**Syntax** `ATS2.DGen.RefVFS (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command: V

**Description** This command sets the Digital Generator relative Volts Full Scale (VFS) reference value. This reference is used for all the Digital Generator relative amplitude units (Vrms, Vp, Vpp, dBu, dBV)

**Example** See example for `ATS2.DGen.Ampl`.

## ATS2.DGen.StepRate

## Property

**Syntax** `ATS2.DGen.StepRate`

**Data Type** Double

**Description** This command sets the rate at which the Digital Generator Special Walking Ones, and Walking Zeros waveform changes state. If the `ATS2.DGen.StepRate` command is set to 5 the Digital Generator will output five words with the same bit pattern and then change to the next bit pattern.

**See Also** `ATS2.DGen.Wfm`

**ATS2.DGen.Wfm****Method**

**Syntax** `ATS2 .DGen .Wfm .Primary .Secondary`

**Part**

Name	Description
<i>Primary</i>	This part defines the basic waveform type.
<i>Secondary</i>	This part defines the basic waveform modifier.

Primary	Secondary	Description
.Sine	.Normal	Sine Normal
	.VarPhase	Sine Var Phase
	.Stereo	Sine Stereo
	.Dual	Sine Dual
	.ShapedBurst	Sine Shaped Burst
	.EQSine	EQ Sine
	.Burst	Sine Burst
	.SinePlusDC	Sine + Offset
.IMD	.SMPTE_4_1	SMPTE/DIN 4:1
	.SMPTE_1_1	SMPTE/DIN 1:1
.Square		Square.
.Noise		Noise
.Arbitrary		Arb Wfm
.Special	.Polarity	Polarity
	.Monotonicity	Monotonicity
	.PassThru	Pass Thru
	.JTest	J-Test
	.WalkingOnes	Walking Ones
	.WalkingZeros	Walking Zeros
	.Constant	Constant Value
.Random	Random	

**Description** This command sets the Digital Generator waveform. The table above shows the possible settings for the `ATS2 .DGen .Wfm` command.

**Example** See example for `ATS2 .DGen .WfmName`.

**ATS2.DGen.WfmName****Property****Syntax**            **ATS2.DGen.WfmName****Data Type**        String                    Long Path and File Names permitted up to 128 characters.**Description**     This command loads the designated arbitrary waveform file (.AGM or .AGS) into the Digital Generator. A Mono waveform (.AGM) is loaded into both the 1 and 2 generator buffers.

Buffer 1 : This buffer is associated with the DSP channel 1.

Buffer 2 : This buffer is associated with the DSP channel 2.

Note: This command can also be used to control the Analog Generator arbitrary waveform file selection.

**See Also**            ATS2.DGen.Wfm**Example**

```

Sub Main
  ATS.Application.NewTest
  ATS.Application.PanelClose(apbAnalogGen)
  ATS.Application.PanelClose(apbAnalogInput)
  ATS.Application.PanelOpen(apbDigitalGen, apbSmall)
  ATS.Application.PanelOpen(apbAnalyzer, apbSmall)
  ATS.Application.PanelOpen(apbDigIO, apbSmall)
  ATS2.Inst.Selection = apbInstFastTestAnalyzer
  ATS2.Inst.FastTest.InputFormat = apbInstInDigital
  ATS2.Dio.InFormat = apbDioInGenMon
  ATS2.DGen.Wfm.Arbitrary
  ATS2.DGen.WfmName = "48kMulTone31.agm"
  ATS2.DGen.OutputOn = True
  ATS.Application.PanelOpen(apbSweep, apbSmall)
  ATS.Sweep.Data(1).Id = 6309
  ATS.Sweep.Data(1).Top("dBFS") = 0.000000
  ATS.Sweep.Source(1).Id = 5621
  ATS.Sweep.Source(1).Steps = 200
  ATS.Sweep.Stereo = True
  ATS.Sweep.Start
  ATS.Graph.OptimizeLeft
End Sub

```



### ATS2.Dio.FlagInvalidRdg

Method

**Syntax**      `ATS2.Dio.FlagInvalidRdg [ (ByVal Channel As Constant) ]`

**Result**      Boolean

<i>True</i>	Error
<i>False</i>	Proper operation

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description**      This reading uses OR logic to determine if either the channel A or channel B Validity bit is set.

When using the optional parameter this reading returns the state of the Validity bit for the specified channel.

The reading is driven directly by the V (Validity) bit defined in the Professional and Consumer standards. One Validity bit is sent in each subframe

**See Also**      `ATS2.Dio.OutSendInvalid`

**Example**      `Sub Main`  
                  `ATS.Application.NewTest`  
                  `ATS.Application.PanelOpen(apbDigIO, apbSmall)`  
                  `ATS2.Dio.InFormat = apbDioInGenMon`  
                  `ATS.Application.PanelOpen(apbDigIO ,apbLarge)`  
                  `ATS2.Dio.OutSendInvalid = True`  
                  `Wait 1`  
                  `If ATS2.Dio.FlagConfidenceRdg = True Then _`

```

    Debug.Print "Confidence Error"
If ATS2.Dio.FlagLockRdg = True Then _
    Debug.Print "Lock Error"
If ATS2.Dio.FlagCodingRdg = True Then _
    Debug.Print "Coding Error"
If ATS2.Dio.FlagParityRdg = True Then _
    Debug.Print "Parity Error"
If ATS2.Dio.FlagInvalidRdg(apbChA) = True Then _
    Debug.Print "ChA Invalid Error"
If ATS2.Dio.FlagInvalidRdg(apbChB) = True Then _
    Debug.Print "ChB Invalid Error"
End Sub

```

---

## ATS2.Dio.FlagCodingRdg

## Method

**Syntax**            **ATS2.Dio.FlagCodingRdg**

**Result**            Boolean

<i>True</i>	Error
<i>False</i>	Proper operation

**Description**    This reading returns the status of the biphas coding for the input serial data stream. The Coding reading indicates a deviation from proper biphas coding in the input serial stream (ignoring preambles). Proper biphas signals can never remain at a logic high or logic low level for more than two consecutive Unit Intervals (UI) except in the preamble. The preamble deliberately deviates from biphas coding in order to provide a unique frame synchronization signal, so preambles are excluded from the function of the Coding indicators.

**Example**            See example for `ATS2.Dio.FlagInvalidRdg`.

---

## ATS2.Dio.FlagConfidenceRdg

## Method

**Syntax**            **ATS2.Dio.FlagConfidenceRdg**

<b>Result</b>	Boolean
	<i>True</i> Error
	<i>False</i> Proper operation
<b>Description</b>	The Confidence reading returns True when the ratio between the amplitude of the three UI long pulse and the following one UI-long pulse in a preamble becomes large enough to cause an increasing probability of errors when slicing the received signal into logic high and low values. This large ratio occurs when the transmission bandwidth has been reduced to marginal or unacceptable values. Under these conditions, selection of hardware input equalization (XLR with EQ or BNC with EQ rather than XLR or BNC selections of the Input Format field) will often compensate for the cable bandwidth reduction, and provide reliable measurements.
<b>Example</b>	See example for <code>ATS2.Dio.FlagInvalidRdg</code> .

---

## ATS2.Dio.FlagLockRdg

## Method

<b>Syntax</b>	<code>ATS2.Dio.FlagLockRdg</code>
<b>Result</b>	Boolean
	<i>True</i> Error
	<i>False</i> Proper operation
<b>Description</b>	The Lock reading indicates when the digital input phase-locked loop is unable to lock to the incoming signal.
<b>Example</b>	See example for <code>ATS2.Dio.FlagInvalidRdg</code>

---

## ATS2.Dio.FlagParityRdg

## Method

<b>Syntax</b>	<code>ATS2.Dio.FlagParityRdg</code>
<b>Result</b>	Boolean
	<i>True</i> Error
	<i>False</i> Proper operation

**Description** The Parity reading indicates a parity error in either subframe. Correct parity is determined by comparing the P (parity) bit with the sum of the remaining 31 bits in each subframe. Any single bit error or odd number of bit errors introduced in transmission within a subframe will cause a Parity error indication, but even numbers of bit errors cannot be detected by this technique.

**Example** See example for `ATS2.Dio.FlagInvalidRdg`

---

## ATS2.Dio.InBitsDisplay

## Property

**Syntax** `ATS2.Dio.InBitsDisplay`

**Data Type** Constant

*apbDioData*

Display Data bits.

*apbDioActive*

Display Active bits.

**Description** This command sets the Digital Input/Output Input Bits Display mode.

---

## ATS2.Dio.InDecode

## Property

**Syntax** `ATS2.Dio.InDecode`

**Data Type** Constant

*apbDioCodeBits*

No data compression applied.

*apbDioCodeU\_Law*

Apply  $\mu$ -Law decoding to data signal.

*apbDioCodeA\_Law*

Apply A-Law decoding to data signal.

**Description** This command selects the Digital Interface Receive Data Format Expansion.

**See Also**      `ATS2.Dio.OutEncode`

**Example**      `Sub Main`  
                   `ATS2.Dio.InDecode = 1      'μ-Law decode`  
                   `End Sub`

---

## ATS2.Dio.InDeEmp

## Property

**Syntax**      `ATS2.Dio.InDeEmp`

**Data Type**    Constant

`apbDioInOff`  
                   Off

`apbDioIn_50`  
                   50/15us 0dB

`apbDioIn_50_12`  
                   50/15us + 12dB

`apbDioIn_J17`  
                   J17 0dB

`apbDioIn_J17_20`  
                   J17 + 20dB

**Description**    This command selects the Digital Input/Output input deemphasis. CD type (50/15 us) or CCITT J17 deemphasis may be selected as desired. Either deemphasis characteristic may be selected with either zero dB insertion loss at low frequencies (0 dB selections in each case) or with a gain factor (+12 dB for 50/15us, +20 dB for J17) to compensate for the matching headroom allowances of the ATS-2 Digital Generator preemphasis `ATS2.Dio.OutPreEmp` capability.

**See Also**      `ATS2.Dio.OutPreEmp`

**Example**      See example for `ATS2.Dio.OutFormat`.

**ATS2.Dio.InFormat****Property****Syntax**      `ATS2.Dio.InFormat`**Data Type**      Constant*apbDioInXLR*

XLR (Bal): Front panel XLR digital input connector, balanced

*apbDioInBNC*

BNC (unbal): Front panel BNC digital input connector, unbalanced

*apbDioInOptical*

Optical: Front panel Toslink optical input connector

*apbDioInGenMon*

Gen Mon: Digital Generator XLR or BNC output connector

*apbDioInDualXLR*

Dual XLR (bal)

**Description**      This command sets the Digital Input/Output Input Format.**Example**      See example for `ATS2.Dio.PeakRdg`**ATS2.Dio.InImpedance****Property****Syntax**      `ATS2.Dio.InImpedance`**Data Type**      Constant

The following list contains the selections relevant to the `ATS2.Dio.InFormat` command XLR (Bal), and Dual XLR (Bal) selections.

*apbDioInHighZ*

High Impedance

*apbDioInLowZ*

110 Ohms

The following list contains the selections relevant to the `ATS2.Dio.InFormat` command BNC (unbal) selection.

*apbDioInHighZ*

High Impedance

*apbDioInLowZ*

75 Ohms

**Description** This command sets the Digital Input/Output Input Impedance based on the selection for the `ATS2.Dio.InFormat` command.

**Example** See example for `ATS2.Dio.PeakRdg`

## ATS2.Dio.InInput

## Property

**Syntax** `ATS2.Dio.InInput`

**Data Type** Constant

*apbDioIn\_I*

Select channel I XLR Input.

*apbDioIn\_II*

Select channel II XLR Input.

**Description** This command selects the Digital Input/Output Input connector. When the Digital Input/Output Input Format is set by the `ATS2.Dio.InFormat` command to any of the XLR selection this command serves as an input switcher to select from one connector (I) or the other (II).

**Example** See example for `ATS2.Dio.InJitterBW`.

## ATS2.Dio.InJitterBW

## Property

**Syntax** `ATS2.Dio.InJitterBW`

**Data Type** Constant

*apbDioIn\_50to100k*

50Hz to 100kHz

```
apbDioIn_120to100k
    120Hz to 100kHz
```

```
apbDioIn_700to100k
    700Hz to 100kHz
```

```
apbDioIn_1200to100k
    1200Hz to 100kHz
```

**Description** This command sets the Digital Input/Output Input bandwidth of the Interface Jitter `ATS2.Dio.JitterRdg` meter.

**Example**

```
Sub Main
    Dim rdg As Double
    ATS2.Dio.OutJitterType = apbDioOutJitterSine
    ATS2.Dio.OutJitterAmpl("UI") = 5.0
    ATS2.Dio.OutJitterFreq("Hz") = 2e3
    ATS2.Dio.InInput = apbDioIn_I
    ATS2.Dio.InJitterMode = apbDioInUI
    ATS2.Dio.InJitterDetector = apbDioInAvg
    ATS2.Dio.InJitterBW = apbDioIn_50to100k
    ATS2.Dio.JitterSettling(5.0, 1e-6, "UI", 1, 0.0,
apbNone)
    ATS2.Dio.JitterTrig
    Do Until ATS2.Dio.JitterReady
        'perform other actions while waiting for reading
        '...
    Loop
    rdg = ATS2.Dio.JitterRdg("UI")
    ATS.Prompt.Text = "Jitter = " & rdg & " UI"
    ATS.Prompt.ShowWithContinue
    Stop
End Sub
```

---

## ATS2.Dio.InJitterDetector

## Property

**Syntax**      `ATS2.Dio.InJitterDetector`

**Data Type**    Constant



*apbDioInPk*

Pk

*apbDioInAvg*

Avg

**Description** This command selects the Digital Input/Output Input detector type for the Interface Jitter `ATS2.Dio.JitterRdg` meter.

**Example** See example for `ATS2.Dio.InJitterBW`.

## ATS2.Dio.InJitterMode

## Property

**Syntax** `ATS2.Dio.InJitterMode`

**Data Type** Constant

*apbDioInUI*

UI: Unit interval

*apbDioInSec*

Sec: Time

**Description** This command selects the measurement mode for the Digital Input/Output Interface Jitter `ATS2.Dio.JitterRdg` meter.

**Example** See example for `ATS2.Dio.InJitterBW`.

## ATS2.Dio.InMonitorMode

## Property

**Syntax** `ATS2.Dio.InMonitorMode`

**Data Type** Constant

*apbDioInPos*

Pos. Peak: causes the Level Monitors to display the most positive value during each measurement interval, which is approximately 1/4 second.

*apbDioInNeg*

Neg. Peak: causes the monitors to display the most negative value during each measurement interval (dBFS)

units cannot be used with the Min mode since the numbers are negative).

*apbDioInPeak*

Abs. Peak: causes display of the absolute value of the largest positive-going or negative-going value during each measurement interval.

*apbDioInHalfPeak*

1/2 Pk-Pk : causes display of the value which is one-half the peak-to-peak range measured during the measurement interval.

**Description** This command sets the Digital Input/Output Peak Monitor

**Example** See example for `ATS2.Dio.PeakRdg`.

## ATS2.Dio.InResolution

## Property

**Syntax** `ATS2.Dio.InResolution`

**Data Type** Long 8 to 24 bits.

**Description** This command sets the Digital Input/Output Input Resolution.

**See Also** `ATS2.Dio.OutResolution`

**Example** See example for `ATS2.Dio.OutFormat`.

## ATS2.Dio.InScaleFreq

## Property

**Syntax** `ATS2.Dio.InScaleFreq`

**Data Type** Constant

*apbDioInOutputRate*

Output Rate: is the Digital Generator output sample rate set by the Rate field near the top of the Output section of the DIO panel.

*apbDioInMeasRate*

Measured Rate: is the input signal sample rate value displayed in the Sample Rate field near the top of the Input section of the DIO panel.

*apbDioInRcvStatBits*

Status Bits A: is the value of sample frequency encoded into the received channel A status bits.

*apbDioInRefRate*

Dio Rate Ref: is the sample rate value defined by the `ATS2.Dio.RefRate` command.

**Description** This command selects a source from which the digital audio sample rate is determined. The frequency of imbedded digital audio signals must be normalized by a digital sample rate before display, whether it is displayed as a numeric frequency counter display or as a frequency component on an FFT graph.

**See Also** `ATS2.Dio.RefRate`

**Example** See example for `ATS2.Dio.OutFormat`.

---

## ATS2.Dio.JitterRdg

## Property

**Syntax** `ATS2.Dio.JitterRdg (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following unit is valid when the <code>ATS2.Dio.InJitterMode</code> command is set to UI (Unit Interval): UI The following unit is valid when the <code>ATS2.Dio.InJitterMode</code> command is set to Sec (Time): sec

**Description** This command returns a settled reading for the Digital Input/Output Interface Jitter meter and zeros the ready count.

**See Also** `ATS2.Dio.JitterReady`, `ATS2.Dio.JitterSettling`, `ATS2.Dio.JitterTrig`, `ATS2.Dio.InJitterMode`

**Example** See example for `ATS2.Dio.InJitterBW`.

---

## ATS2.Dio.JitterReady

**Property**

**Syntax** `ATS2.Dio.JitterReady`

**Data Type** Integer

0 Reading not ready.  
>0 Reading ready.

**Description** This command returns the Digital Input/Output Interface Jitter meter meter settled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the `ATS2.Dio.JitterRdg` or `ATS2.Dio.JitterTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.Dio.JitterRdg` command will be guaranteed to return quickly.

**See Also** `ATS2.Dio.JitterRdg`, `ATS2.Dio.JitterSettling`, `ATS2.Dio.JitterTrig`

**Example** See example for `ATS2.Dio.InJitterBW`.

---

## ATS2.Dio.JitterSettling

**Method**

**Syntax** `ATS2.Dio.JitterSettling` (ByVal *Tolerance* As Double, ByVal *Floor* As Double, ByVal *FloorUnit* As String, ByVal *Points* As Integer, ByVal *Delay* As Double, ByVal *Algorithm* As Constant)

**Parameter** See Appendix A for Settling Algorithm and parameter name descriptions.

**Description** This command sets the settling parameters for the `ATS2.Dio.JitterRdg` command.

**See Also** `ATS2.Dio.JitterRdg`, `ATS2.Dio.JitterReady`,  
`ATS2.Dio.JitterTrig`

**Example** See example for `ATS2.Dio.JitterBW`.

## ATS2.Dio.JitterTrig

**Method**

**Syntax** `ATS2.Inst.`

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Dio.JitterRdg` command. The reading in progress is aborted.

**See Also** `ATS2.Dio.JitterRdg`, `ATS2.Dio.JitterReady`,  
`ATS2.Dio.JitterSettling`

**Example** See example for `ATS2.Dio.InJitterBW`.

## ATS2.Dio.OutEncode

**Property**

**Syntax** `ATS2.Dio.OutEncode`

**Data Type** Constant

*apbDioCodeBits*

No data compression applied.

*apbDioCodeU\_Law*

Apply  $\mu$ -Law encoding to data signal.

*apbDioCodeA\_Law*

Apply A-Law encoding to data signal.

**Description** This command selects the Digital Interface Transmit Data Format Compression.

**See Also** `ATS.S2CDio.OutEncode`

**Example** Sub Main  
`ATS2.Dio.OutEncode = apbDioCodeU_Law`  
 End Sub

**ATS2.Dio.OutFormat****Property****Syntax**      **ATS2.Dio.OutFormat****Data Type**    Constant*apbDioOutXLR*

XLR

*apbDioOutBNC*

BNC

*apbDioOutOptical*

Optical

*apbDioOutDualXLR\_OSR*

Dual XLR

*apbDioOutDualXLR\_2XOSR*

Dual XLR 2xOSR

**Description**    This command sets the Digital Input/Output Output source.**Example**

Const INTERVU\_AMPL As Integer = 6053

Const INTERVU\_TIME As Integer = 5612

Sub Main

ATS.Application.NewTest

    With **ATS.S2CDio**        **.OutFormat** = apbDioOutXLR        **.OutRate** ("Hz") = 48000        **.OutResolution** = 20        **.OutPreEmp** = apbDioOutOff        **.InDeEmp** = apbDioInOff        **.InFormat** = apbDioInGenMon        **.InResolution** = 20        **.InScaleFreq** = apbDioInMeasRate

End With

ATS2.Inst.Selection = apbInstIntervuAnalyzer

ATS2.Inst.Intervu.WfmDisplay = \_

apbIntervuInterpolate

ATS2.Trigger.Source = apbTrigRcvChASub

With ATS.Sweep

```

        .Source(1).Id = INTERVU_TIME
        .Source(1).Start("sec") = 0.0
        .Source(1).Stop("sec") = 4e-6
        .Source(1).Steps = 255
        .Data(1).Id = INTERVU_AMPL
        .Data(1).Top("V") = 3.0
        .Data(1).Bottom("V") = -3.0
        .Start
    End With
End Sub

```

---

## ATS2.Dio.OutInvert

## Property

**Syntax**            **ATS2.Dio.OutInvert**

**Data Type**        Boolean

*True*                Invert output.  
*False*               Normal non-inverting output.

**Description**     This command sets the Digital Output to normal polarity or inverted polarity (180 degrees out of phase).

---

## ATS2.Dio.OutJitterAmpl

## Property

**Syntax**            **ATS2.Dio.OutInvert** (ByVal *Unit* As String)

**Data Type**        Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: UI, dBUI, sec.

**Description**     This command sets the Digital Input/Output Jitter Amplitude value.

**See Also**         `ATS2.Dio.OutJitterEqCurve`

**Example**          See example for `ATS2.Dio.InJitterBW`.

## ATS2.Dio.OutJitterEqCurve

### Method

**Syntax** `ATS2.Dio.OutJitterEqCurve (ByVal FileName As String, ByVal Column As Integer)`

Parameter	Name	Description
	<i>FileName</i>	Any valid DOS path and file name. The file must be an ATS Eq file (.atsq).
	<i>Column</i>	0 = Source 1 settings. 1 = Data 1 measurements. 2 = Data 2 measurements. 3 = Data 3 measurements. 4 = Data 4 measurements. 5 = Data 5 measurements. 6 = Data 6 measurements. 7 = Source 2 settings.

**Result** Boolean

*True* File attachment successful.

*False* File attachment failed.

**Description** This command attaches a Eq file to the Jitter Generator. Values in the file will be used as multiply factors in calculating the Digital Input/Output Jitter Amplitude value.

**See Also** `ATS2.Dio.OutJitterAmpl`

## ATS2.Dio.OutJitterFreq

### Property

**Syntax** `ATS2.Dio.OutJitterFreq (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Hz

**Description** This command sets the Digital Input/Output Jitter Frequency value.

**Example** See example for `ATS2.Dio.InJitterBW`.



---

**ATS2.Dio.OutJitterType****Property****Syntax**      **ATS2.Dio.OutJitterType****Data Type**      Constant*apbDioOutJitterOff*

Off

*apbDioOutJitterSine*

Sine

*apbDioOutJitterEQSine*

EQ Sine

**Description**      This command sets the type of jitter that may be added to the digital output signal at the XLR, BNC, and optical outputs to test the ability of a digital device to reject input jitter.**Example**      See example for `ATS2.Dio.InJitterBW`.

---

**ATS2.Dio.OutParityError****Property****Syntax**      **ATS2.Dio.OutParityError****Data Type**      Boolean*True*

Generate Parity Error in digital output.

*False*

Normal output.

**Description**      This command sets the Digital Output Parity Bit to produce a Parity Error or to normal operation.**See Also**      `ATS2.Dio.FlagParityRdg`**Example**      See example for `ATS2.Dio.FlagInvalidRdg`.

---

**ATS2.Dio.OutPreEmp****Property****Syntax**      **ATS2.Dio.OutPreEmp****Data Type**      Constant

```
apbDioOutOff
    Off
apbDioOut_50
    50/15us 0dB
apbDioOut_50_12
    50/15us - 10dB
apbDioOut_J17
    J17 0dB
apbDioOut_J17_20
    J17 - 20dB
```

**Description** This command selects the Digital Input/Output Output Preemphasis. CD type (50/15 us) or CCITT J17 deemphasis may be selected as desired. Either preemphasis function may be selected at normal gain or with a headroom allowance. When program material is put through a preemphasis function, the natural high-frequency roll-off of most music and voice signals and typical practices of headroom allowance for peaks are sufficient to assure that high-frequency signals will not clip (exceed digital full scale). However, full-scale test signals such as sinewave sweeps or multitone signals with equal amplitude at all frequencies will clip at high frequencies when preemphasis is applied. To prevent this clipping due to the high-frequency boost, two additional selections are available which automatically attenuate the signal level sufficiently to provide headroom at the highest frequencies. These headroom allowances are selected by the 50/15 us -10 dB and J17 -20 dB choices. Each will attenuate the audio signal by the specified amount, which is slightly greater than the boost at the maximum possible audio frequency for the chosen preemphasis characteristic. If desired, a matching deemphasis with gain selection is available in the Deemphasis field or via the `ATS2.Dio.InDeEmp` command of the Input section of the DIO panel to provide an overall unity gain and flat response during digital domain stimulus/response measurements.

**See Also** `ATS2.Dio.InDeEmp`

**Example** See example for `ATS2.Dio.OutFormat`.

---

## ATS2.Dio.OutRate

**Property**

<b>Syntax</b>	<b>ATS2.Dio.OutRate</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	The digital output sample rate of ATS may be freely set across the range from 28.8 kHz to 108 kHz.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, and dPPM.
<b>Description</b>	This command sets the Digital Output Sample Rate Frequency value.	
<b>Example</b>	See example for <code>ATS2.Dio.OutFormat</code> .	

---

## ATS2.Dio.OutResolution

**Property**

<b>Syntax</b>	<b>ATS2.Dio.OutResolution</b>	
<b>Data Type</b>	Integer	The width or resolution of the imbedded digital audio signal may be set to any value from 8 to 24 bits.
<b>Description</b>	<p>This command sets the Digital Output Resolution.</p> <p>Internally, the imbedded digital audio signal is always generated at 24 bits. When any smaller value is selected in the Resolution field, the 24-bit word is rounded (not truncated) to the specified value and dither is added (unless disabled) at the proper amplitude for the value entered. Bits below the value entered in the Resolution field are set to zero. The output resolution is independent from the input resolution.</p>	
<b>See Also</b>	<code>ATS2.Dio.InResolution</code>	
<b>Example</b>	See example for <code>ATS2.Dio.OutFormat</code> .	

---

## ATS2.Dio.OutScaleFreq

**Property**

<b>Syntax</b>	<b>ATS2.Dio.OutScaleFreq</b>
---------------	------------------------------

<b>Data Type</b>	Constant
	<p><i>apbDioOutOutputRate</i></p> <p>Output Rate: the value in the Sample Rate-OSR field near the top of the Output section of the DIO panel.</p> <p><i>apbDioOutMeasRate</i></p> <p>Measured Rate: is the input signal sample rate value displayed in the Sample Rate field near the top of the Input section of the DIO panel.</p> <p><i>apbDioOutRefRate</i></p> <p>Dio Rate Ref: is the sample rate value defined by the <code>ATS2.Dio.RefRate</code> command.</p>
<b>Description</b>	This command sets the source for scaling for the audio frequency embedded in the digital output signal.
<b>See Also</b>	<code>ATS2.Dio.RefRate</code>
<b>Example</b>	<pre>Sub Main   ATS2.Dio.OutScaleFreq = 1   'use measured input rate End Sub</pre>

---

## ATS2.Dio.OutSendInvalid

## Property

**Syntax**      `ATS2.Dio.OutSendInvalid`

**Data Type**      Boolean

<i>True</i>	Set
<i>False</i>	Clear validity bit.

**Description**      This command sets or clears the validity bit.

The AES/EBU and Consumer standards define a data invalid bit for each subframe. This is the V bit of the VUCP bits (validity, user, channel status, parity). Actual usage of this bit is not totally standardized, but a common usage in digital tape recorders (for example) is to set this bit as invalid if the tape is not moving and valid if the tape is playing. ATS permits the user to simultaneously set both

channel A and B validity bits as true (check the Send Invalid box) or false (Send Invalid box unchecked) in order to test whether and how digital devices respond to the bit.

## ATS2.Dio.OutVoltage

## Property

**Syntax** `ATS2.Dio.Voltage (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command: Vpp

**Description** This command sets the amplitude of the serial pulse train at the XLR, BNC and optical outputs, which may be used to simulate cable attenuation.

**Example** See example for `ATS2.Dio.VoltageRdg`.

## ATS2.Dio.PeakRdg

## Property

**Syntax** `ATS2.Dio.PeakRdg (ByVal Channel As Constant, ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits.

**Description** This command returns a unsettled reading for the Digital Input/Output Peak Monitor meter and zeros the ready count.

**See Also** `ATS2.Dio.PeakReady`, `ATS2.Dio.PeakTrig`

**Example**

```
Sub Main
    Dim rdgA As Double, rdgB As Double
```

```

ATS.Application.
ATS2.Dio.InFormat = apbDioInGenMon
ATS2.Dio.InImpedance = apbDioInHighZ
ATS2.Dio.InMonitorMode = apbDioInPeak
ATS2.Dio.PeakTrig(apbChA)
ATS2.Dio.PeakTrig(apbChB)
Do Until (ATS2.Dio.PeakReady(apbChA) And _
ATS2.Dio.PeakReady(apbChB))
    'perform other actions while
    'waiting for readings ...
Loop
= ATS2.Dio.PeakRdg(apbChA, "FFS")
= ATS2.Dio.PeakRdg(apbChB, "FFS")
ATS.Prompt.Text = "Ch A = " & & " FFS" & _
    Chr(13) & "Ch B = " & rdgB & " FFS"
ATS.Prompt.
Stop
End Sub

```

## ATS2.Dio.PeakReady

## Property

**Syntax**      **ATS2.Dio.PeakReady** (ByVal *Channel* As Constant)

**Data Type**      Integer

0                      Reading not ready.

>0                     Reading ready.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description**      This command returns the Digital Input/Output Peak Monitor meter unsettled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any

number of times. Only a call to the `ATS2.Dio.PeakRdg` command will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.Dio.PeakRdg` command will be guaranteed to return quickly.

**See Also** `ATS2.Dio.PeakRdg`, `ATS2.Dio.PeakTrig`

**Example** See example for `ATS2.Dio.PeakRdg`.

---

## ATS2.Dio.PeakTrig

## Method

**Syntax** `ATS2.Dio.PeakTrig` (ByVal *Channel* As Constant)

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Dio.PeakRdg` command. The reading in progress is aborted.

**See Also** `ATS2.Dio.PeakRdg`, `ATS2.Dio.PeakReady`

**Example** See example for `ATS2.Dio.PeakRdg`.

---

## ATS2.Dio.RateRdg

## Property

**Syntax** `ATS2.Dio.RateRdg` (ByVal *Unit* As String)

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, and dPPM.

**Description** This command returns a settled reading for the Digital Input/Output Sample Rate meter and zeros the ready count.

**See Also** `ATS2.Dio.RateReady`, `ATS2.Dio.RateSettling`,  
`ATS2.Dio.RateTrig`

**Example**

```

Sub Main
    Dim rdgA As Double
    ATS2.Dio.InImpedance = apbDioInHighZ
    ATS2.Dio.InFormat = apbDioInGenMon

    ATS2.Dio.RateSettling(5.0, 100e-3, "Hz", 1, 0.0,
apbNone)
    ATS2.Dio.RateTrig
    Do Until ATS2.Dio.RateReady
        'perform other actions while
        'waiting for readings ...
    Loop
    rdgA = ATS2.Dio.RateRdg("Hz")
    ATS.Prompt.Text = "Ch A = " & rdgA & " Hz"
    ATS.Prompt.ShowWithContinue
    Stop
End Sub

```

---

## ATS2.Dio.RateReady

## Property

**Syntax** `ATS2.Dio.RateReady`

**Data Type** Integer

0 Reading not ready.  
>0 Reading ready.

**Description** This command returns the Digital Input/Output Sample Rate meter settled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the `ATS2.Dio.RateRdg` or `ATS2.Dio.RateTrig` commands will zero the ready count.



If the reading is found to be ready, a call to the `ATS2.Dio.RateRdg` command will be guaranteed to return quickly.

**See Also** `ATS2.Dio.RateRdg`, `ATS2.Dio.RateSettling`,  
`ATS2.Dio.RateTrig`

**Example** See example for `ATS2.Dio.RateRdg`.

---

## ATS2.Dio.RateSettling

## Method

**Syntax** `ATS2.Dio.RateSettling`(ByVal *Tolerance* As Double,  
ByVal *Floor* As Double, ByVal *FloorUnit* As  
String, ByVal *Points* As Integer, ByVal *Delay* As  
Double, ByVal *Algorithm* As Constant)

**Parameter** See Appendix A for Settling Algorithm and parameter name descriptions.

**Description** This command sets the settling parameters for the `ATS2.Dio.RateRdg` command.

**See Also** `ATS2.Dio.RateRdg`, `ATS2.Dio.RateReady`,  
`ATS2.Dio.RateTrig`

**Example** See example for `ATS2.Dio.RateRdg`.

---

## ATS2.Dio.RateTrig

## Method

**Syntax** `ATS2.Inst.`

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Dio.RateRdg` command. The reading in progress is aborted.

**See Also** `ATS2.Dio.RateRdg`, `ATS2.Dio.RateReady`,  
`ATS2.Dio.RateSettling`

**Example** See example for `ATS2.Dio.RateRdg`.

## ATS2.Dio.RefRate

## Property

<b>Syntax</b>	<b>ATS2.Dio.RefRate</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	Set Reference Sample Rate value.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Hz
<b>Description</b>	This command sets the Digital Input/Output Sample Rate reference.	
<b>See Also</b>	ATS2.Inst.InScaleFreq, ATS2.Inst.OutRate, ATS2.Inst.RateRdg	

## ATS2.Dio.VoltageRdg

## Property

<b>Syntax</b>	<b>ATS2.Dio.VoltageRdg</b> (ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command: Vpp
<b>Description</b>	This command returns a settled reading for the Digital Input/Output Voltage meter and zeros the ready count.	
<b>See Also</b>	ATS2.Dio.VoltageReady, ATS2.Dio.VoltageSettling, ATS2.Dio.VoltageTrig	
<b>Example</b>	<pre> Sub Main     ATS2.Dio.InFormat = apbDioInGenMon     <b>ATS2.Dio.VoltageSettling</b>(5.0, .001, "Vpp", 1, _         0.0, apbNone)     ATS2.Dio.OutVoltage("Vpp") = 1.0     <b>ATS2.Dio.VoltageTrig</b>     Do Until <b>ATS2.Dio.VoltageReady</b>         'perform other actions while         'waiting for readings ...     Loop     rdgA = <b>ATS2.Dio.VoltageRdg</b>("Vpp") </pre>	

```

    ATS.Prompt.Text = "Ch A = " & rdgA & " Vpp"
    ATS.Prompt.ShowWithContinue
    Stop
End Sub

```

---

## ATS2.Dio.VoltageReady

Property

**Syntax**      `ATS2.Dio.VoltageReady`

**Data Type**    Integer

0                  Reading not ready.  
 >0                Reading ready.

**Description**    This command returns the Digital Input/Output Voltage meter settled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the `ATS2.Dio.VoltageRdg` or `ATS2.Dio.VoltageTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.Dio.VoltageRdg` command will be guaranteed to return quickly.

**See Also**        `ATS2.Dio.VoltageRdg`, `ATS2.Dio.VoltageSettling`,  
`ATS2.Dio.VoltageTrig`

**Example**        See example for `ATS2.Dio.VoltageRdg`.

---

## ATS2.Dio.VoltageSettling

Method

**Syntax**        `ATS2.Dio.VoltageSettling (ByVal Tolerance As Double, ByVal Floor As Double, ByVal FloorUnit As String, ByVal Points As Integer, ByVal Delay As Double, ByVal Algorithm As Constant)`

<b>Parameter</b>	See Appendix A for Settling Algorithm and parameter name descriptions.
<b>Description</b>	This command sets the settling parameters for the <code>ATS2.Dio.VoltageRdg</code> command.
<b>See Also</b>	<code>ATS2.Dio.VoltageRdg</code> , <code>ATS2.Dio.VoltageReady</code> , <code>ATS2.Dio.VoltageTrig</code>
<b>Example</b>	See example for <code>ATS2.Dio.VoltageRdg</code> .

---

## ATS2.Dio.VoltageTrig

## Method

<b>Syntax</b>	<code>ATS2.Inst.</code>
<b>Description</b>	Causes a restart of the reading cycle and zeros the ready count for the <code>ATS2.Dio.VoltageRdg</code> command. The reading in progress is aborted.
<b>See Also</b>	<code>ATS2.Dio.VoltageRdg</code> , <code>ATS2.Dio.VoltageReady</code> , <code>ATS2.Dio.VoltageSettling</code>
<b>Example</b>	See example for <code>ATS2.Dio.VoltageRdg</code> .

### ATS2.Hardware.AdjustmentDate

Method

**Syntax**            `ATS2.Hardware.AdjustmentDate`

**Result**            String

**Description**      This command returns the ASCII text string for the ATS-2 Adjustment Date.

**Example**            Sub Main  
                      Debug.Print "Serial Number " & \_  
                                          ATS2.Hardware.SerialNumber  
                      Debug.Print "Adjustment Date " & \_  
                                          ATS2.Hardware.AdjustmentDate  
                      Debug.Print "Board Serial Number " & \_  
                                          ATS2.Hardware.BoardSerialNumber  
                      Debug.Print "Manufacture Date " & \_  
                                          ATS2.Hardware.ManufactureDate  
                      If ATS2.Hardware.OptionInstalled(apbDCX127) Then \_  
                                          Debug.Print "DCX-127 Option Installed"  
                      If ATS2.Hardware.OptionInstalled(apbGPIB) Then \_  
                                          Debug.Print "GPIB Option Installed"  
                      If ATS2.Hardware.OptionInstalled(apbHighBandwidth) \_  
                                          Then Debug.Print "High Bandwidth Option \_  
                                          Installed"  
                      If ATS2.Hardware.OptionInstalled(apbHwSwitcher) \_  
                                          Then Debug.Print "Switcher Option Installed"  
                      If ATS2.Hardware.OptionInstalled(apbInputZ600) \_  
                                          Then Debug.Print "600 InputZ Option Installed"  
                      If ATS2.Hardware.OptionInstalled(apbIntervu) Then \_  
                                          Debug.Print "Interv Option Installed"  
                      If ATS2.Hardware.OptionInstalled(apbOutputZ150) \_  
                                          Then Debug.Print "150 OutputZ Option Installed"

```

    If ATS2.Hardware.OptionInstalled(apbOutputZ200) _
        Then Debug.Print "200 OutputZ Option Installed"
    If ATS2.Hardware.OptionInstalled(apbOutputZ600) _
        Then Debug.Print "600 OutputZ Option Installed"
End Sub

```

**Output**

```

Serial Number 100001
Adjustment Date 1/28/2002
Board Serial Number -62-1-0
Manufacture Date 1/24/2002
DCX-127 Option Installed
High Bandwidth Option Installed
Switcher Option Installed
Interv Option Installed
150 OutputZ Option Installed

```

**ATS2.Hardware.BoardSerialNumber****Method**

<b>Syntax</b>	<code>ATS2.Hardware.BoardSerialNumber</code>
<b>Result</b>	String
<b>Description</b>	This command returns the ASCII text string for the ATS-2 Board Serial Number.
<b>Example</b>	See example for <code>ATS2.Hardware.AdjustmentDate</code> .

**ATS2.Hardware.ManufactureDate****Method**

<b>Syntax</b>	<code>ATS2.Hardware.ManufactureDate</code>
<b>Result</b>	String
<b>Description</b>	This command returns the ASCII text string for the ATS-2 ManufactureDate.
<b>Example</b>	See example for <code>ATS2.Hardware.AdjustmentDate</code> .

**ATS2.Hardware.OptionInstalled****Method**

**Syntax** `ATS2.Hardware.OptionInstalled(ByVal Option As Constant)`

**Data Type** Boolean

*True* Option Installed.  
*False* Option NOT Installed.

**Parameter****Name** **Description**

Name	Description
<i>Option</i>	apbDCX127 = DCX-127 apbGPIB = GPIB Interface apbHighBandwidth = High Bandwidth apbHwSwitcher = Switcher apbIntervu = apbInputZ600 = 600 Ohm Input Impedance apbOutputZ150 = 150 Ohm Output Impedance apbOutputZ200 = 200 Ohm Output Impedance apbOutputZ600 = 600 Ohm Output Impedance

**Description** This command returns the status of the specified hardware option.

**Example** See example for `ATS2.Hardware.AdjustmentDate`.

**ATS2.Hardware.SerialNumber****Method**

**Syntax** `ATS2.Hardware.SerialNumber`

**Result** String

**Description** This command returns the ASCII text string for the ATS-2 Serial Number.

**Example** See example for `ATS2.Hardware.AdjustmentDate`.





# Chapter 26

## Audio Analyzer

---

### ATS2.Inst.Analyzer.DCCoupled

Property

**Syntax** `ATS2.Inst.Analyzer.DCCoupled (ByVal Channel As Constant)`

**Data Type** Boolean

<i>True</i>	DC Coupled
<i>False</i>	Not DC Coupled

**Parameter**

Name	Description
<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** This command enables ATS to DC couple the input to the Audio Analyzer. The level meter can then be used to measure DC.

**Example** See example for `ATS2.Inst.Analyzer.FreqRdg`.

---

### ATS2.Inst.Analyzer.FreqRdg

Property

**Syntax** `ATS2.Inst.Analyzer.FreqRdg (ByVal Channel As Constant, ByVal Unit As String)`

**Data Type** Double

**Parameter**

Name	Description
<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
<i>Unit</i>	The following units are available: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.

**Description** This command returns a settled reading for the Audio Analyzer Frequency meter and zeros the ready count.

**See Also**     ATS2.Inst.Analyzer.FreqReady,  
                   ATS2.Inst.Analyzer.FreqSettling,  
                   ATS2.Inst.Analyzer.FreqTrig

**Example**

```

Sub Main
    ATS.Application.NewTest
    ATS2.AGen.Output = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS.Application.PanelOpen(apbAnalyzer, apbLarge)
    With ATS2.Inst.Analyzer
        .InputFormat = apbInstInAnalog
        .DCCoupled(apbChA) = False
        .RangeAuto(apbChA) = False
        .Range(apbChA, "FFS") = 1.0
        .LevelSettling(apbChA, 1.0, 0.000001, _
            "V", 3, 0.03, apbExponential)
        .FreqSettling(apbChA, 0.5, 0.01, _
            "Hz", 3, 0.03, apbExponential)
        .LevelTrig(apbChA)
        .FreqTrig(apbChA)
    Do
        Loop Until .LevelReady(apbChA) And _
            .FreqReady(apbChA)
        var1 = .LevelRdg(apbChA, "V")
        var2 = .FreqRdg(apbChA, "Hz")
    End With
    Text1$= "Channel A Level " & _
        Str$(Format(var1, "##.000")) & "V"
    Text2$= "Channel A Frequency " & _
        Str$(Format(var2, "##.000")) & "Hz"
    ATS.Prompt.Text = Text1$ & Chr(13) & Text2$
    ATS.Prompt.ShowWithContinue
    Stop
End Sub

```

## ATS2.Inst.Analyzer.FreqReady

## Property

**Syntax**     **ATS2.Inst.Analyzer.FreqReady** (ByVal *Channel* As  
                   Constant)

**Data Type** Integer

0 Reading not ready.  
>0 Reading ready.

**Parameter**

Name	Description
<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description**

This command returns the Audio Analyzer Frequency meter settled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the `ATS2.Inst.Analyzer.FreqRdg` or `ATS2.Inst.Analyzer.FreqTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.Inst.Analyzer.FreqRdg` command will be guaranteed to return quickly.

**See Also**

`ATS2.Inst.Analyzer.FreqRdg`,  
`ATS2.Inst.Analyzer.FreqSettling`,  
`ATS2.Inst.Analyzer.FreqTrig`

**Example**

See example for `ATS2.Inst.Analyzer.FreqRdg`.

---

## ATS2.Inst.Analyzer.FreqSettling

**Method**

**Syntax**

**ATS2.Inst.Analyzer.FreqSettling**(ByVal *Channel* As Constant, ByVal *Tolerance* As Double, ByVal *Floor* As Double, ByVal *FloorUnit* As String, ByVal *Points* As Integer, ByVal *Delay* As Double, ByVal *Algorithm* As Constant)

**Parameter**

See Appendix A for Settling Algorithm and parameter name descriptions.

- Description** This command sets the settling parameters for the `ATS2.Inst.Analyzer.FreqRdg` command.
- See Also** `ATS2.Inst.Analyzer.FreqRdg`,  
`ATS2.Inst.Analyzer.FreqReady`,  
`ATS2.Inst.Analyzer.FreqTrig`
- Example** See example for `ATS2.Inst.Analyzer.FreqRdg`.

---

## ATS2.Inst.Analyzer.FreqTrig

Method

**Syntax** `ATS2.Inst.Analyzer.FreqTrig`(ByVal *Channel* As Constant)

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Inst.Analyzer.FreqRdg` command. The reading in progress is aborted.

**See Also** `ATS2.Inst.Analyzer.FreqRdg`,  
`ATS2.Inst.Analyzer.FreqReady`,  
`ATS2.Inst.Analyzer.FreqSettling`

**Example** See example for `ATS2.Inst.Analyzer.FreqRdg`.

---

## ATS2.Inst.Analyzer.FuncBPBRFreq

Property

**Syntax** `ATS2.Inst.Analyzer.FuncBPBRFreq`(ByVal *Unit* As String)

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	The following units are available: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.

**Description** This command sets the Audio Analyzer BandPass/BandReject filter to the frequency value passed.

The Bandpass filter affects only the main (Reading) meter, not the Level or Frequency reading. It is a highly-selective filter of about 1/13 octave bandwidth (Q=19, 3 dB bandwidth about 5.2% of center frequency). The bandpass filter is tunable across the audio spectrum from 0.04% to 42% of the sample rate (20 Hz to 20 kHz at a 48 kHz sample rate). It is used in Bandpass and Crosstalk functions.

The Bandreject (notch) function of the filter is used in the two THD+N functions. It is tunable from 0.04% to 42% of the sample rate (20 Hz to 20 kHz at a 48 kHz rate).

**See Also** `ATS2.Inst.Analyzer.FuncBPBRTuning`,  
`ATS2.Inst.Analyzer.FuncMode`

### Example

```
Sub Main
  ATS.Application.NewTest
  ATS2.AGen.Output = True
  ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
  ATS.Application.PanelOpen(apbAnalyzer, apbLarge)
  With ATS2.Inst.Analyzer
    .InputFormat = apbInstInAnalog
    .FuncMode = apbAnlrBandpass
    .RdgRate = apbAnlr32PerSec
    .FuncDetector = apbAnlrRMS
    .FuncBPBRTuning = apbAnlrFixed
    .FuncBPBRFreq("Hz") = 999.99
    .FuncSettling(apbChA, 3.0, 1.0e-8, _
      "V", 3, 0.1, apbExponential)
    .FuncTrig(apbChA)
  Do
  Loop Until .FuncReady(apbChA)
  var1 = .FuncRdg(apbChA, "V")
  End With
  Text$= "Bandpass Amplitude " & Str$(Format(var1, _
    "##.000")) & "V"
  ATS.Prompt.Text = Text$ & Chr(13)
  ATS.Prompt.ShowWithContinue
  Stop
End Sub
```

**ATS2.Inst.Analyzer.FuncBPBRTuning****Property****Syntax**      `ATS2.Inst.Analyzer.FuncBPBRTuning`**Data Type**      Constant*apbAnlrCounterTuned*

Counter Tuned: the frequency value measured by the ANALYZER Frequency counter is the filter steering source. This function would be selected when making THD+N or Crosstalk measurements from an external signal such as reproduction of a Compact Disc or digital audio tape or reception of a digital signal from a distant source.

*apbAnlrSweepTrack*

Sweep Track: the filter tracks the frequency of whichever generator is selected in the Source 1 or Source 2 fields of the Sweep panel.

*apbAnlrAGenTrack*

AGen Track: the digital bandpass-bandreject filter tracks the frequency of the Analog Generator, This mode is useful for testing A/D converters driven from ATS-2's analog output.

*apbAnlrDGenTrack*

DGen Track: the filter will automatically track the frequency of the Digital Generator. This mode would normally be used when sweeping digital input-digital output devices with stimulus coming from ATS-2's Digital Generator.

*apbAnlrFixed*

Fixed: the filter will be fixed at the frequency entered in the BP/BR Filter Freq field just below unless the filter is being deliberately varied as part of a sweep test. To sweep the filter frequency during a test, select BP/BR Filter Freq as the Source 1 or Source 2 parameter on the Sweep panel. Fixed tuning mode must be selected in order to use the BP/BR Filter Freq parameter as a Source value.

**Description**      This command sets the Audio Analyzer Bandpass Bandreject filter Tuning source.**See Also**      `ATS2.Inst.Analyzer.FuncBPBRFreq`

**Example** See example for `ATS2.Inst.Analyzer.FuncBPBRFreq`.

## ATS2.Inst.Analyzer.FuncDetector

## Property

**Syntax** `ATS2.Inst.Analyzer.FuncDetector`

**Data Type** Constant

*apbAnlrRMS*

RMS: Level and Function meters use RMS detector.

*apbAnlrFastRMS*

Fast RMS: Level and Function meters use Fast RMS detector.

*apbAnlrQpeak*

Qpeak: Level meters use RMS detection and Function meters use Qpeak detector.

**Description** This command selects the Audio Analyzer Detector type for the Level and Function meters.

**See Also** `ATS2.Inst.Analyzer.RdgRate`,  
`ATS2.Inst.Analyzer.FuncMode`,  
`ATS2.Inst.Analyzer.FuncRange`,  
`ATS2.Inst.Analyzer.FuncRangeAuto`

**Example**

```
Sub Main
  ATS.Application.NewTest
  ATS2.AGen.OutputOn = True
  ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
  ATS.Application.PanelOpen(apbAnalyzer, apbLarge)
  With ATS2.Inst.Analyzer
    .InputFormat = apbInstInAnalog
    .FuncMode = apbAnlrTHDNRatio
    .FuncRangeAuto(apbChA) = True
  ' .Range(apbChA, "dBFS") = 0.0
    .RdgRate = apbAnlrAuto
    .FuncDetector = apbAnlrRMS
    .FuncFilterHP = apbAnlrHP10
    .FuncFilterLP = apbAnlrLP20k
```

```

    .FuncFilter = apbAnlrNone
    .FuncSettling(apbChA, 3.0, 0.000010, _
        "%", 3, 0.1, apbExponential)
    Wait .5
    .FuncTrig(apbChA)
    Do
        Loop Until .FuncReady(apbChA)
    var = .FuncRdg(apbChA, "%")
    End With
    Text$= "THD+N = " & Str$(Format(var, "##.00000")) _
        & "%"
    ATS.Prompt.Text = Text$ & Chr(13)
    ATS.Prompt.ShowWithContinue
    Stop
End Sub

```

## ATS2.Inst.Analyzer.FuncFilter

## Property

**Syntax**      **ATS2.Inst.Analyzer.FuncFilter**

**Data Type**      Constant

The following list contains the selections relevant to the `ATS2.Inst.Analyzer.FuncMode` command Amplitude, THD+N Ampl, THD+N Ratio, and 2-Ch Ratio selections.

```

apbAnlrNone
    None

apbAnlrAWeight
    "A" Weighting

apbAnlrCCIR
    CCIR Weighting

apbAnlrFWeight
    F Weighting

apbAnlrCCITT
    CCITT Weighting

```



*apbAnlrCMessage*  
C-Message Weighting

*apbAnlrHarmonicWeight*  
Ho-2 Harmonic Weighting (THD Only)

*apbAnlrUserWeight*  
User Weighting Filter

The following list contains the selections relevant to the `ATS2.Inst.Analyzer.FuncMode` command Bandpass selection.

*apbAnlrNarrow*  
Narrow

*apbAnlrNarrowX2*  
Narrow, Freq x2

*apbAnlrNarrowX3*  
Narrow, Freq x3

*apbAnlrNarrowX4*  
Narrow, Freq x4

*apbAnlrNarrowX5*  
Narrow, Freq x5

The following list contains the selections relevant to the `ATS2.Inst.Analyzer.FuncMode` command Crosstalk selection.

*apbAnlrNarrow*  
Narrow

- Description** This command selects the Audio Analyzer Weighting Filter.
- See Also** `ATS2.Inst.Analyzer.FuncFilterHP`,  
`ATS2.Inst.Analyzer.FuncFilterLP`
- Example** See example for `ATS2.Inst.Analyzer.FuncDetector`.

## ATS2.Inst.Analyzer.FuncFilterFilename Get Only Property

<b>Syntax</b>	<code>c</code>	
<b>Data Type</b>	Integer	Column number.
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Number</i>	Sweep Data selection number (1-6)
<b>Description</b>	This command sets the column number used for the Lower Limit of a Sweep Data.	
<b>See Also</b>	ATS.Sweep.Data.AutoDiv, ATS.Sweep.Data.LogLin	
<b>Example</b>	See example for <code>ATS.Sweep.Append</code> .	

## ATS2.Inst.Analyzer.FuncFilterHP Property

<b>Syntax</b>	<code>ATS2.Inst.Analyzer.FuncFilterHP</code>	
<b>Data Type</b>	Constant	
	<i>apbAnlrHP10</i>	<10 Hz
	<i>apbAnlrHP22</i>	22 Hz
	<i>apbAnlrHP100</i>	100 Hz
	<i>apbAnlrHP400</i>	400 Hz
	<i>apbAnlrHPUser</i>	High Pass user defined filter
<b>Description</b>	This command selects the Audio Analyzer High Pass filter used with the function meters.	
<b>See Also</b>	<code>ATS2.Inst.Analyzer.FuncFilterLP</code>	
<b>Example</b>	See example for <code>ATS2.Inst.Analyzer.FuncDetector</code> .	

**ATS2.Inst.Analyzer.FuncFilterLP****Property****Syntax** `ATS2.Inst.Analyzer.FuncFilterLP`**Data Type** Integer`apbAnlrLPFS2`

Fs/2

`apbAnlrLP20K`

20 kHz LP

`apbAnlrLP15K`

15 kHz LP

`apbAnlrLPUser`

Low Pass user defined filter

**Description** This command selects the Audio Analyzer Low Pass filter used with the function meters.**See Also** `ATS2.Inst.Analyzer.FuncFilterHP`**Example** See example for `ATS2.Inst.Analyzer.FuncDetector`.**ATS2.Inst.Analyzer.FuncFilterUserDefined****Method****Syntax** `ATS2.Inst.Analyzer.FuncFilterHP (ByVal FilterType as Constant, ByVal FileName as String)`**Parameter****Name****Description***FilterType*

`apbAnlrUserFilterHP` = Specifies that the file defined by the second parameter (*FileName*) will be attached as a high pass filter. The high pass used defined filter can be selected by using the `ATS2.Inst.Analyzer.FuncFilterHP` command. The file must be an ATS high-pass filter file (.afh) for this selection.

`apbAnlrUserFilterLP` = Specifies that the file defined by the second parameter (*FileName*) will be attached as a low pass filter. The high pass used defined filter can be selected by using the

		ATS2.Inst.Analyzer.FuncFilterLP command. The file must be an ATS low-pass filter file (.afl) for this selection.
		apbAnlrUserFilterWeight = Specifies that the file defined by the second parameter ( <i>FileName</i> ) will be attached as a weighting filter. The high pass used defined filter can be selected by using the
		ATS2.Inst.Analyzer.FuncFilter command. The file must be an ATS weighting filter file (.afw) for this selection.
	<i>FileName</i>	Long Path and File Names permitted up to 128 characters. Enter "None" for the file name to remove the User-Defined Filter.
<b>Result</b>	Boolean	
	<i>True</i>	File attachment successful.
	<i>False</i>	File attachment failed.
<b>Description</b>	This command attaches a User-Defined Filter for use with the Audio Analyzer. When a filter is attached, the design is tested to determine that there are not more than 2 second-order sections used to create the filter and that the filter is stable. If either of these conditions is not met then this command returns False.	
<b>See Also</b>	ATS2.Inst.Analyzer.FuncFilterHP, ATS2.Inst.Analyzer.FuncFilterLP, ATS2.Inst.Analyzer.FuncFilter	
<b>Example</b>	<pre>Sub Main     <b>ATS2.Inst.Analyzer.FuncFilterUserDefined</b> _         (apbAnlrUserFilterHP, "My_HP-1.afh") End Sub</pre>	

---

## ATS2.Inst.Analyzer.FuncMode

## Property

<b>Syntax</b>	ATS2.Inst.Analyzer.FuncMode
<b>Data Type</b>	Constant

*apbAnlrAmplitude*

Amplitude: Amplitude mode measurements can differ from the Level meter measurements due to two factors:

Amplitude mode measurements are affected by the high-pass (`ATS2.Inst.Analyzer.FilterHP`), low-pass (`ATS2.Inst.Analyzer.FilterLP`), and weighting filter (`ATS2.Inst.Analyzer.FuncFilter`) commands, while Level meter readings are unfiltered. Amplitude measurements may be made with the quasi-peak or one of the RMS detectors, while the Level meters always use the same type of RMS detector selected with the `ATS2.Inst.Analyzer.FuncDetector` command.

*apbAnlrTwoChRatio*

2-Channel Ratio: 2-Ch Ratio mode displays in the Function Reading meter display the amplitude ratio between the Function Reading meter channel and the alternate channel. Both Level meters continue to display the absolute level on each channel. 2-Ch Ratio function is useful while adjusting stereo channel amplitudes to match or for measuring gain or loss when the analyzer inputs are connected at the input and output of a device.

*apbAnlrCrossTalk*

Crosstalk: Crosstalk mode is identical to the 2-Ch Ratio mode except that the tunable bandpass filter is also engaged in the main (Function Reading) meter before the measurement. Crosstalk mode will thus provide more accurate measurements of low-amplitude signals in the presence of noise, since most wide-band noise will be rejected by the filter. The filter must be tuned to the frequency of the signal on the driven channel.

*apbAnlrTHDNRatio*

THD+N Ratio: The THD+N mode uses a bandreject (notch) filter to remove the fundamental sinewave signal so that the detector may measure the remaining harmonic distortion products and noise. The THD+N Ratio mode expresses the

distortion product and noise amplitudes relative to the amplitude of the unfiltered signal measured by the Level meter. Units of % and dB (below fundamental) are commonly used in THD+N Ratio function. THD+N Ratio is used much more commonly than THD+N Amplitude, but in an amplitude sweep THD+N Ratio appears to show increasing distortion and noise with decreasing signal amplitude because the distortion and noise is stated as a ratio to the decreasing signal. THD+N Amplitude may be more useful for amplitude sweeps. The bandreject filter center frequency may be fixed or may track one of several other parameters.

#### *apbAnlrTHDNABS*

THD+N Ampl: The THD+N Amplitude mode uses a bandreject (notch) filter to remove the fundamental sinewave signal so that the detector may measure the remaining harmonic distortion products and noise. The THD+N Ampl (amplitude) mode expresses amplitude of the remaining distortion products and noise in absolute units (FFS, %FS, dBFS, bits with digital signals; Volts, dBV, dBu, etc. with analog signals), independent of the amplitude of the fundamental signal. THD+N Ampl mode is particularly useful when performing amplitude sweeps of audio devices, since it helps make clear that the noise component is (typically) a constant amplitude unrelated to the signal amplitude. THD+N Ratio in an amplitude sweep obscures this fact, since the measured distortion and noise appears to increase with decreasing signal amplitude because it is being stated as a ratio to the decreasing signal.

The bandreject filter center frequency may be fixed or may track one of several other parameters.

#### *apbAnlrBandpass*

Bandpass: Bandpass mode is a selective voltmeter. It includes a narrow bandpass filter of about 1/13 octave (Q=19, 3 dB bandwidth about 5.2% of center frequency). The bandpass filter center frequency may be fixed or may track one of several other parameters. The filter may be

tuned to the steering source fundamental frequency or to the 2nd, 3rd, 4th, or 5th harmonic of the tuning source. This harmonic tracking ability permits swept measurements of individual harmonic distortion, limited to a maximum value of 42% of the sample rate:

*apbAnlrSMPTE*

SMPTE/DIN: The SMPTE/DIN mode is designed to function with two-tone intermodulation distortion test signals complying with the SMPTE or DIN standard, or similar to these. These signals normally consist of a low-frequency tone between 40 Hz and 500 Hz combined with a high-frequency tone above 2 kHz. Typical SMPTE tone combinations are 60 Hz and 7 kHz, while 250 Hz and 8 kHz are often used with the DIN standard. The amplitude ratio of the LF tone to the HF tone is commonly 4:1 but the tones are sometimes at equal amplitudes. The analyzer function measures the amplitude of the sidebands near the high-frequency tone and expresses their amplitude as a ratio to the HF tone amplitude.

*apbAnlrPhase*

Phase: The Phase mode measures the phase of the B channel signal relative to the A channel signal (B-A), expressing the result in degrees. This corresponds to the phase measurement technique of the hardware analog analyzer, and to the Sine Variable Phase waveform of both the analog and digital generators. The Channel B display field is removed when the Phase function is selected.

**Description** This command selects the analysis mode of the Audio Analyzer Function meters.

**Example** See example for `ATS2.Inst.Analyzer.FuncDetector`.

---

## ATS2.Inst.Analyzer.FuncPhaseMode

## Property

**Syntax** `ATS2.Inst.Analyzer.FuncPhaseMode`

**Data Type** Constant

```

apbAnlrPhaseAuto
    Auto
apbAnlrPhase180
    -180 +180 deg
apbAnlrPhase360
    0 +360 deg
apbAnlrPhase270
    -90 +270 deg

```

**Description** This function sets the Audio Analyzer Function meter Phase measurement range.

---

## ATS2.Inst.Analyzer.FuncRange

## Property

**Syntax** `ATS2.Inst.Analyzer.FuncRange (ByVal Channel As Constant, ByVal Unit As String)`

**Data Type** Double

The following values are the range boundaries for the dB unit: 0.000, -6.021, -12.041, -18.062, -24.82, -30.103, -36.124, -42.144, -48.165, -54.185, -60.206, -66.227, -72.247, -78.268, -84.288, -90.309

If an arbitrary value between the range boundaries is entered the next higher range will be selected.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Unit</i>	The following units are available: X/Y, dB

**Description** This command sets the Audio Analyzer Function meter Range.

**See Also** `ATS2.Inst.Analyzer.RdgRate`,  
`ATS2.Inst.Analyzer.FuncMode`,  
`ATS2.Inst.Analyzer.FuncRangeAuto`

**Example** See example for `ATS2.Inst.Analyzer.FuncDetector`.



**ATS2.Inst.Analyzer.FuncRangeAuto****Property**

<b>Syntax</b>	<code>ATS2.Inst.Analyzer.FuncRangeAuto</code>
<b>Data Type</b>	Boolean
	<i>True</i> Auto range.
	<i>False</i> Fixed range.
<b>Description</b>	This command sets the Audio Analyzer Function meter to Auto or Fixed Range.
<b>See Also</b>	<code>ATS2.Inst.Analyzer.RdgRate</code> , <code>ATS2.Inst.Analyzer.FuncMode</code> , <code>ATS2.Inst.Analyzer.FuncRange</code>
<b>Example</b>	See example for <code>ATS2.Inst.Analyzer.FuncDetector</code> .

**ATS2.Inst.Analyzer.FuncRdg****Property**

**Syntax**            `ATS2.Inst.Analyzer.FuncRdg` (ByVal *Channel* As Constant, ByVal *Unit* As String)

**Data Type**        Double

**Parameter**

Name	Description
<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
<i>Unit</i>	String that designates the desired unit.

The following units are valid for the "Digital" selection of the `ATS2.Inst.Analyzer.InputFormat` command for the Amplitude, THD+N Ampl, and Bandpass Function meter Modes: FFS, %FS, dBFS, Bits, V, Vp, Vpp, dBu, dBV, dBr A, dBr B.

The following units are valid for the "Analog" selection of the `ATS2.Inst.Analyzer.InputFormat` command for the Amplitude, THD+N Ampl, and Bandpass Function meter Modes: V, dBu, dBV, dBr A, dBr B, dBg A, dBg B, W.

The following units (% , dB, PPM, X/Y) are available for the following Function meter Modes: 2-Ch Ratio, Crosstalk, THD+N Ratio, SMPTE IMD.

The (deg) units is available for the Function meter Phase Mode.

**Description** This command returns a settled reading from the Audio Analyzer Function meter and zeros the ready count.

**See Also** `ATS2.Inst.Analyzer.FuncMode`,  
`ATS2.Inst.Analyzer.FuncReady`,  
`ATS2.Inst.Analyzer.FuncSettling`,  
`ATS2.Inst.Analyzer.FuncTrig`

**Example** See example for `ATS2.Inst.Analyzer.FuncDetector`.

---

## ATS2.Inst.Analyzer.FuncReady

## Property

**Syntax** `ATS2.Inst.Analyzer.FuncReady (ByVal Channel As Constant)`

**Data Type** Integer

0 Reading not ready.  
>0 Reading ready.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** This command returns the Audio Analyzer Function meter settled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the `ATS2.Inst.Analyzer.FuncRdg` or

`ATS2.Inst.Analyzer.FuncTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.Inst.Analyzer.FuncRdg` command will be guaranteed to return quickly.

**See Also** `ATS2.Inst.Analyzer.FuncRdg`,  
`ATS2.Inst.Analyzer.FuncSettling`,  
`ATS2.Inst.Analyzer.FuncTrig`

**Example** See example for `ATS2.Inst.Analyzer.FuncDetector`.

---

## ATS2.Inst.Analyzer.FuncSettling

**Method**

**Syntax** `ATS2.Inst.Analyzer.FuncSettling`(ByVal *Channel* As Constant, ByVal *Tolerance* As Double, ByVal *Floor* As Double, ByVal *FloorUnit* As String, ByVal *Points* As Integer, ByVal *Delay* As Double, ByVal *Algorithm* As Constant)

**Parameter** See Appendix A for Settling Algorithm and parameter name descriptions.

**Description** This command sets the settling parameters for the `ATS2.Inst.Analyzer.FuncRdg` command.

**See Also** `ATS2.Inst.Analyzer.FuncRdg`,  
`ATS2.Inst.Analyzer.FuncReady`,  
`ATS2.Inst.Analyzer.FuncTrig`

**Example** See example for `ATS2.Inst.Analyzer.FuncDetector`.

---

## ATS2.Inst.Analyzer.FuncTrig

**Method**

**Syntax** `ATS2.Inst.Analyzer.FuncTrig`

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Inst.Analyzer.FuncRdg` command. The reading in progress is aborted.

**See Also** `ATS2.Inst.Analyzer.FuncRdg`,  
`ATS2.Inst.Analyzer.FuncReady`,  
`ATS2.Inst.Analyzer.FuncSettling`

**Example** See example for `ATS2.Inst.Analyzer.FuncDetector`.

---

## ATS2.Inst.Analyzer.InputFormat

Property

**Syntax** `ATS2.Inst.Analyzer.InputFormat`

**Data Type** Constant  
`apbInstInDigital`  
 Digital  
`apbInstInAnalog`  
 Analog

**Description** This command sets the Audio Analyzer Input Format.

**Example** See example for `ATS2.Inst.Analyzer.FuncDetector`.

---

## ATS2.Inst.Analyzer.LevelRdg

Property

**Syntax** `ATS2.Inst.Analyzer.LevelRdg (ByVal Channel As Constant, ByVal Unit As String)`

**Data Type** Double

**Description** This command returns a settled reading for the Audio Analyzer Level meter and zeros the ready count.

Parameter	Name	Description
	<code>Channel</code>	apbChA = Select channel A apbChB = Select channel B
	<code>Unit</code>	String that designates the desired unit. The following units are valid for the "Digital" selection of the <code>ATS2.Inst.Analyzer.InputFormat</code> command: FFS, %FS, dBFS, Bits, V, Vp, Vpp, dBu, dBV,

dBr 1, dBr 2.

The following units are valid for the "Analog" selection of the `ATS2.Inst.Analyzer.InputFormat` command: V, dBu, dBV, dBr A, dBr B, dBg A, dBg B, and W.

**See Also** `ATS2.Inst.Analyzer.LevelReady`,  
`ATS2.Inst.Analyzer.LevelSettling`,  
`ATS2.Inst.Analyzer.LevelTrig`

**Example** See example for `ATS2.Inst.Analyzer.FreqRdg`.

## ATS2.Inst.Analyzer.LevelReady

## Property

**Syntax** `ATS2.Inst.Analyzer.LevelReady (ByVal Channel As Constant)`

**Data Type** Integer

<i>0</i>	Reading not ready.
<i>&gt;0</i>	Reading ready.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** This command returns the Audio Analyzer Level Monitor meter settled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the `ATS2.Inst.Analyzer.LevelRdg` or `ATS2.Inst.Analyzer.LevelTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.Inst.Analyzer.LevelRdg` command will be guaranteed to return quickly.

**See Also** `ATS2.Inst.Analyzer.LevelRdg`,  
`ATS2.Inst.Analyzer.LevelSettling`,  
`ATS2.Inst.Analyzer.LevelTrig`

**Example** See example for `ATS2.Inst.Analyzer.FreqRdg`.

---

## ATS2.Inst.Analyzer.LevelSettling

### Method

**Syntax** `ATS2.Inst.Analyzer.LevelSettling (ByVal Channel As Constant, ByVal Tolerance As Double, ByVal Floor As Double, ByVal FloorUnit As String, ByVal Points As Integer, ByVal Delay As Double, ByVal Algorithm As Constant)`

**Parameter** See Appendix A for Settling Algorithm and parameter name descriptions.

**Description** This command sets the settling parameters for the `ATS2.Inst.Analyzer.LevelRdg` command.

**See Also** `ATS2.Inst.Analyzer.LevelRdg`,  
`ATS2.Inst.Analyzer.LevelReady`,  
`ATS2.Inst.Analyzer.LevelTrig`

**Example** See example for `ATS2.Inst.Analyzer.FreqRdg`.

---

## ATS2.Inst.Analyzer.LevelTrig

### Method

**Syntax** `ATS2.Inst.Analyzer.LevelTrig (ByVal Channel As Constant)`

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Inst.Analyzer.LevelRdg` command. The reading in progress is aborted.

**See Also** `ATS2.Inst.Analyzer.LevelRdg`,  
`ATS2.Inst.Analyzer.LevelSettling`,  
`ATS2.Inst.Analyzer.LevelTrig`

**Example** See example for `ATS2.Inst.Analyzer.FreqRdg`.

---

## ATS2.Inst.Analyzer.Range

## Property

**Syntax** `ATS2.Inst.Analyzer.Range (ByVal Channel As Constant, ByVal Unit As String)`

**Data Type** Double

The following values are the range boundaries for the dB unit: 0.000, -6.021, -12.041, -18.062, -24.82, -30.103, -36.124, -42.144, -48.165, -54.185, -60.206, -66.227, -72.247, -78.268, -84.288, -90.309

If an arbitrary value between the range boundaries is entered the next higher range will be selected.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Unit</i>	The following units are available: FFS, %FS, dBFS, dBr A or dBr B for channel B.

**Description** This command sets the Audio Analyzer input Range and returns the nominal full scale of the range in use.

**See Also** `ATS2.Inst.Analyzer.RangeAuto`

**Example** See example for `ATS2.Inst.Analyzer.FreqRdg`.

---

## ATS2.Inst.Analyzer.RangeAuto

## Property

**Syntax** `ATS2.Inst.Analyzer.RangeAuto`

**Data Type** Boolean

<i>True</i>	Auto range.
<i>False</i>	Fixed range.

**Description** This command sets the Audio Analyzer Function meter to Auto or Fixed Range. Care must be taken when using Fixed range that the input signal does not exceed the selected range.

**See Also** `ATS2.Inst.Analyzer.Range`

**Example** See example for `ATS2.Inst.Analyzer.FreqRdg.`

---

## ATS2.Inst.Analyzer.RdgRate

## Property

**Syntax** `ATS2.Inst.Analyzer.RdgRate`

**Data Type** Constant

*apbAnlrAuto*

Auto: this selection manages selection of the reading rate as a function of the frequency being measured and the instrument function so as to provide rapid testing speeds along with sufficient integration for accuracy at the present test frequency.

*apbAnlr4PerSec*

4/Sec

*apbAnlr8PerSec*

8/Sec

*apbAnlr16PerSec*

16/Sec

*apbAnlr32PerSec*

32/Sec

*apbAnlr64PerSec*

64/Sec

*apbAnlr128PerSec*

128/Sec

*apbAnlr256PerSec*

256/sec

**Description** This command sets the Audio Analyzer Reading update Rate (integration period) for all of the Audio Analyzer meters.



**See Also**      `ATS2.Inst.Analyzer.FuncDetector`,  
                  `ATS2.Inst.Analyzer.FuncMode`

**Example**      See example for `ATS2.Inst.Analyzer.FuncDetector`.

## User Notes

# Chapter 27

## Multitone Audio Analyzer

### ATS2.Inst.FastTest.FreqRes

### Property

**Syntax** `ATS2 . Inst . FastTest . FreqRes (ByVal Unit As String)`

**Data Type** Double Valid amplitude settings are from +/- 0.0 to 13.0 %.

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command: %

**Description** This command sets the Multitone Audio Analyzer Frequency Resolution.

The Frequency Resolution field is a numeric entry field with % units. The user may enter values up to 13% which are used in Response and Distortion Measurement functions.

In Response function, the amplitudes of all FFT bins within plus and minus the Frequency Resolution value of each sweep table value are combined in RSS (root-sum-square) fashion and furnished to the computer as the integrated amplitude of the bins within that range. The purpose of this function is to provide accurate frequency response measurements of devices with wow and flutter. Wow and flutter spreads the energy from a single tone across a narrow spectral band.

In Distortion function, the amplitudes of all FFT bins within plus and minus the Frequency Resolution value of each sweep table value are excluded from the RSS computation of energy falling between tones. Distortion function defines all signals other than the fundamental tones as distortion and noise. Entering a non-zero value of Frequency Resolution causes flutter sidebands to not be included in the distortion measurement.

**Example** See example for `ATS2.Inst.FastTest.TransformLength`.

---

## ATS2.Inst.FastTest.InputFormat

**Property**

**Syntax** `ATS2.Inst.FastTest.InputFormat`

**Data Type** Integer

*apbInstInDigital*

Digital

*apbInstInAnalog*

Analog

**Description** This command sets the Multitone Audio Analyzer Input Format.

**Example** See example for `ATS2.Inst.FastTest.TransformLength`.

---

## ATS2.Inst.FastTest.Mode

**Property**

**Syntax** `ATS2.Inst.FastTest.Mode`

**Data Type** Constant

*apbFastTestSpectrum*

Spectrum: Provides a normal FFT spectrum display with no processing except for peak picking. The Spectrum selection is typically used without a sweep table (.ADS file), and with a relative large number of Steps at Source 1 of the Sweep panel to provide good frequency resolution. Typical Steps values are from 250 to 500. If the transform length results in more FFT bins between the Start-Stop frequency span than are being plotted, peak-picking takes place. With peak-picking, the DSP searches all FFT bins between the previous plotted point and the point presently being plotted and sends the highest bin amplitude in that range as the amplitude of the new point to be sure that no signals are missed.

*apbFastTestResponse*

Response: is always used with a sweep table (.ADS file) listing the exact frequencies of the sinewaves in the multitone signal to be used for frequency response measurements. The DSP returns to the computer for plotting only the amplitudes of the FFT bins containing those exact frequencies, resulting in a frequency response graph.

If the value in the Frequency Resolution field is greater than zero, the DSP performs an RSS (root-sum-square) integration of all the bin amplitudes within plus or minus the Frequency Resolution value around each sweep table frequency and sends the integrated sum value to the computer to be plotted. This mode is intended for frequency response measurements on devices such as analog tape recorders which introduce frequency modulation (flutter) to signals. Flutter spreads each tone's energy across a small region of the spectrum. This reduces the amplitude of the fundamental tone, since the total energy in the fundamental and all sidebands remains constant during frequency modulation. The RSS summation combines this spread energy back into a single value, much as the human hearing system responds to signals with small amounts of FM.

*apbFastTestDistortion*

Distortion: excludes the amplitudes of the FFT bins known (from the generator waveform) to contain fundamental signals. All other bin amplitudes are summed (RSS) between each adjacent pair of frequencies requested from the DSP by the computer. It is not necessary to use a sweep table (.ADS file) listing the fundamental frequencies of the sinewaves in the multitone signal being used. Distortion and noise can thus be summed across spans determined by the Sweep panel Start, Stop, Log/Lin, and number of Steps, or the spans can be determined by a sweep table. If it is desired to sum the noise and distortion into critical bands, a sweep table can be used which defines the edges of the human hearing system critical bands. The resulting distortion and noise curve is normally compared to

the composite masking curve generated in Masking function.

If the value in the Frequency Resolution field is greater than zero, the DSP also excludes all the bin amplitudes within plus or minus the Frequency Resolution value around each sweep table frequency before sending the integrated sum value to the computer to be plotted. This mode is intended for distortion measurements on devices such as analog tape recorders which introduce frequency modulation (flutter) to signals. Flutter spreads each tone's energy across a small region of the spectrum. If these close-in sidebands which fall outside the bin containing the fundamental are not to be measured as distortion, they must be excluded, much as the human hearing system masks low amplitude signals nearby in frequency to a stronger signal.

#### *apbFastTestNoise*

Noise: this selection may be used with a sweep table (.ADS file) listing the fundamental frequencies of the multitone signal in use, but need not be. Noise mode depends on the FASTTEST Transform length being set to the value twice the length of the waveform file which generates the multitone signal. The analyzer frequency resolution is thus twice the resolution of the generated signal. The result is that every alternate analyzer FFT bin falls between bins at which the generated signal could contain fundamentals or bins into which harmonic or intermodulation distortion products due to the generated signal fundamental signals could fall (assuming that the device under test does not shift fundamental frequencies or produce frequency modulation). The amplitudes of these alternate empty bins consists of noise generated in the device under test, largely unaffected by fundamental signals or distortion. If the same sweep table is used in Noise mode that is used for response and distortion measurements, the resulting graph will be a spectrum analysis of noise in the presence of test signal. If a two-point sweep is made with Start at 20 Hz and Stop at

20 kHz, for example, the plotted value at 20 kHz represents the RSS integration of all empty bins across the audio band.

#### *apbFastTestMasking*

Masking: this selection generates a composite masking curve for the particular multitone signal in use. The shape of the curves is based on a model published by psychacoustician Brian Moore in the Proceedings of the AES 12th International Conference, June 1993, pp 22-23. The shape of the curves varies with frequency. The center frequency of each section of the composite masking curve is located at the fundamental frequencies present in the waveform file downloaded to the generator buffer. The reference amplitude at each frequency is determined by the measured amplitude at each fundamental frequency. The masking curve is normally used by saving it as a limit (.ADL) file, then comparing a Distortion function curve (usually with critical band spacing) to that limit curve.

#### *apbFastTestCrosstalk*

Crosstalk: depends upon the multitone test signal having one or more unique tone frequencies on each stereo channel, in addition to any number of tones which are common to both channels. Crosstalk function determines which generator frequencies are unique to a channel and measures the amplitude of the corresponding FFT frequency bin on the opposite channel. Unique frequencies are typically created in multitone signals at frequencies above 500 Hz, where the generator resolution is less limiting and where a bin occupied for crosstalk measurement purposes represents a small portion of the total bins for measurement of total integrated noise and distortion across that portion of the spectrum. In order to measure crosstalk in both directions (from A to B and from B to A), it is common to insert unique tones at pairs of nearby frequencies on each channel. For example, if monaural signals (tones on both channels) exist at about 500 Hz and 640 Hz, a crosstalk-measurement tone might be inserted at 560 Hz on Channel A and at 575 Hz on Channel B. Crosstalk is commonly used with a sweep table

corresponding to the approximate frequencies where the pairs of crosstalk frequencies have been inserted. At each frequency in the sweep table, the DSP will report the amplitude of the crosstalk-containing bin nearest the requested frequency. The FASTTEST Channel 1 curve will show measurements of crosstalk into that frequency from Channel 2, and vice-versa. If the stereo channels have been mistakenly reversed, the crosstalk measurements will show the levels of the tones in the channel on which they were transmitted. This makes it easy to automatically determine cases of swapped channels by setting an upper limit file for each channel.

**Description** This command sets the Multitone Audio Analyzer measurement mode. The `ATS2.Inst.FastTest.Mode` command controls the type of post-processing done to FFT results before they are sent to the computer for display and possible limits comparison.

**Example** See example for `ATS2.Inst.FastTest.TransformLength`.

---

## ATS2.Inst.FastTest.PhaseDisplay

## Property

**Syntax** `ATS2.Inst.FastTest.PhaseDisplay`

**Data Type** Constant

*apbFastTestIndependent*

Independent

*apbFastTestInterChannel*

Interchannel

**Description** This command sets the Multitone Audio Analyzer Phase Display mode selection.

The FFT of FASTTEST computes both magnitude and phase arrays as a function of frequency. The phase of coherent signals, such as multitone signals, may be plotted for either or both channels by selecting FASTTEST as the instrument and Ch 1 Phase or Ch 2 Phase as the parameter in the Data browser of the Sweep panel. A sweep table (.ADS file) listing the fundamental signals would be used



in this mode. When the channel 2 Phase Display is selected as Independent, the Ch 1 and Ch 2 Phase parameters each show the absolute phase of the fundamental tones.

It is also possible to plot the interchannel phase difference of stereo signals with FASTTEST. Selecting Interchannel causes the DSP to compute the phase difference between the Ch 1 and Ch 2 Phase signals at each sweep table value and report that computed value to the computer as the Ch 2 Phase parameter. The Ch 1 Phase parameter is unaffected by the Interchannel setting and plots absolute phase of the channel 1 signal.

**Example** See example for `ATS2.Inst.FastTest.TransformLength`.

---

## ATS2.Inst.FastTest.Processing

## Property

**Syntax** `ATS2.Inst.FastTest.Processing`

**Data Type** Constant

*apbFastTestSync*

Synchronous: Normal operation of FASTTEST involves acquisition of a multitone signal which was generated from a multitone waveform file by ATS's Digital Generator. The multitone waveform files furnished with ATS are created so as to be synchronous with one or another of the analyzer acquisition buffer lengths available in FASTTEST. Every sinewave in the generated signal goes through an exact integer number of cycles in the generator buffer and in the analyzer transform buffer. No windowing function is required and maximum theoretical FFT selectivity is achieved with full dynamic range available in bins adjacent to a bin containing a full-scale signal.

*apbFastTestFreqCorrect*

Freq Corrected: A key feature of FASTTEST is its ability to compare the tone frequencies in an acquired multitone waveform with the digital reference copy of the transmitted or pre-recorded waveform presently in the Digital Generator buffers. If this comparison shows that the tone frequencies

have been shifted up or down due to the signal originating from a device with a different clock frequency from the analyzer or due to analog tape player speed errors, FASTTEST corrects all the tone frequencies to the reference signal values. This re-creates the original synchronous relationship so that no window function is required before the FFT, and maximum theoretical FFT selectivity is obtained. The maximum frequency difference which can be corrected is +/-3%. FASTTEST is normally operated with Frequency Error Correction enabled when analyzing signals generated by another Audio Precision instrument or previously recorded and now being reproduced. This mode of operation is selected by the Freq Corrected selection in the Processing field.

#### *apbFastTestWindowed*

Windowed: If for some reason it is desired to measure remotely-generated or pre-recorded signals without use of the Frequency Error Correction feature, it will normally be necessary to use the Hann window function to obtain useful results. The Windowed selection of the Processing field enables the Hann window.

**Description** This command sets the Multitone Audio Analyzer processing.

**Example** See example for `ATS2.Inst.FastTest.TransformLength`.

---

## ATS2.Inst.FastTest.Rdg

## Property

**Syntax** `ATS2.Inst.FastTest.Rdg (ByVal Channel As Constant, ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits.

**Description** This command returns a unsettled reading for the Multitone Audio Analyzer Peak Monitor meter and zeros the ready count.

**See Also** `ATS2.Inst.FastTest.Ready`,  
`ATS2.Inst.FastTest.Trig`

**Example**

```

Sub Main
    ATS.File.OpenTest "FASTTSTC.ATS2"
    With ATS2.Inst.FastTest
        .Trig(apbChA)
    Do
        Loop Until .Ready(apbChA)
        AReading = .Rdg(apbChA, "dBFS")
    End With
    ATS.Prompt.Text = "ChA Peak Mon " _
        & Left(Str$(AReading), 6) & "dBFS"
    ATS.Prompt.ShowWithContinue
    Stop
End Sub
    
```

---

## ATS2.Inst.FastTest.Ready

## Property

**Syntax** `ATS2.Inst.FastTest.Ready (ByVal Channel As Constant)`

**Data Type** Integer

<i>0</i>	Reading not ready.
<i>&gt;0</i>	Reading ready.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** This command returns the Multitone Audio Analyzer Peak Monitor meter unsettled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any

number of times. Only a call to the `ATS2.Inst.FastTest.Rdg` or `ATS2.Inst.FastTest.Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.Inst.FastTest.Rdg` command will be guaranteed to return quickly.

**See Also** `ATS2.Inst.FastTest.Rdg`, `ATS2.Inst.FastTest.Trig`

**Example** See example for `ATS2.Inst.FastTest.Rdg`.

---

## ATS2.Inst.FastTest.TransformLength

## Property

**Syntax** `ATS2.Inst.FastTest.TransformLength`

**Data Type** Constant

*apbFastTestAuto*

Auto: the Auto selection will automatically set the acquisition buffer and transform length to be exactly twice the length of any generator waveform loaded into the Digital Generator buffer. This condition is necessary for the Noise function of FASTTEST to work.

*apbFastTest512*

512

*apbFastTest1k*

1024

*apbFastTest2k*

2048

*apbFastTest4k*

4096

*apbFastTest8k*

8192

*apbFastTest16k*

16384

```
apbFastTest32k
32768
```

**Description** This command sets the Multitone Audio Analyzer FFT Length.

The FFT Length field value of the FASTTEST program controls the record length used as input to the FFT process when either F9/Go or ATS.Sweep.Start is initiated to acquire and transform, or the F6 or ATS.Sweep.Retransform function key or Sweep Transform Data without Acquire menu command is used to re-transform any portion of a record previously acquired. Longer transform lengths produce greater frequency resolution in the resulting FFT, but require longer times to acquire and transform the signal.

### Example

```
Sub Main
ATS.File.OpenTest "FasttstB.ATS2"
With ATS2.Inst.FastTest
    .InputFormat = apbInstInAnalog
    .Mode = apbFastTestSpectrum
    .FreqRes("%") = 1
    .TransformLength = apbFastTest16k
    .Processing = apbFastTestSync
    .TrigSource = apbFastTestDGen
    .TrigDelay("sec") = 0
    .PhaseDisplay = apbFastTestIndependent
ATS.Sweep.Start
ATS.Sweep.Source(1).Table("FASTTST.ATSS", 0)
    .Mode = apbFastTestResponse
ATS.Sweep.Reprocess
    .Mode = apbFastTestDistortion
ATS.Sweep.Reprocess
    .Mode = apbFastTestNoise
ATS.Sweep.Reprocess
    .Mode = apbFastTestMasking
ATS.Sweep.Reprocess
End With
End Sub
```

## ATS2.Inst.FastTest.Trig

### Method

**Syntax** `ATS2.Inst.FastTest.Trig (ByVal Channel As Constant)`

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Inst.FastTest.Rdg` command. The reading in progress is aborted.

**See Also** `ATS2.Inst.FastTest.Rdg`, `ATS2.Inst.FastTest.Ready`

**Example** See example for `ATS2.Inst.FastTest.Rdg`.

## ATS2.Inst.FastTest.TrigDelay

### Property

**Syntax** `ATS2.Inst.FastTest.TrigDelay (ByVal Unit As String)`

**Data Type** Double Values up to 1.365 seconds may be entered.

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command: sec.

**Description** This command sets the Multitone Audio Analyzer trigger delay.

When testing audio transmission paths which include audio processors (compressors, limiters, etc.), it may be desirable to make measurements after the processors have stabilized following any change of level resulting between the multitone burst and the preceding program material. The Trigger Delay field controls the interval between initial recognition of the incoming multitone signal and capture of the portion of signal which will finally be analyzed for response, distortion, noise, etc. Use of any non-zero Trigger Delay requires that the duration of multitone burst transmitted be increased by the same amount over normal minimum burst length.

**Example** See example for `ATS2.Inst.FastTest.TransformLength`.

---

**ATS2.Inst.FastTest.TrigSource****Property**

**Syntax**            **ATS2 . Inst . FastTest . TrigSource**

**Data Type**        Constant

*apbFastTestDGen*

DGen: If the Digital Generator is generating a signal from a waveform file, a Digital Generator trigger is issued each time the first sample from the file is generated.

*apbFastTestTight*

Tight: See description below.

*apbFastTestNormal*

Normal: See description below.

*apbFastTestLoose*

Loose: See description below.

*apbFastTestExternal*

External: If is the signal connected to pin 3 of the 15-pin D-sub connector on the rear of the DSP module. If pin 3 is high (or open circuit, in which case it is pulled high by an internal pull-up resistor), triggering occurs at the next digital sample. Pulling pin 3 low from an external device holds off triggering, with acquisition being triggered on the next sample after pin 3 is pulled high.

*apbFastTestOff*

Off: this selection produces untriggered or free-running operation. Acquisition and processing begins as soon as the F9 key, Go, or `ATS . Sweep . Start` command is initiated. The Off selection is the recommended triggering mode when ATS and FASTTEST are testing devices by simultaneously driving their input and measuring their output as opposed to capturing a pre-recorded or remotely-originated multitone signal.

*apbFastTestAGen*

AGen: If the Analog Generator is generating a signal from a waveform file, a Analog Generator trigger is issued each time the first sample from the file is generated.

**Description** This command sets the Multitone Audio Analyzer Triggering.

1. Using the tone frequencies represented in the Digital Generator buffer as a reference, FASTTEST looks at the received signal to see if the amplitude at each of a majority of those frequencies is within an acceptable relative amplitude range of the corresponding component of the reference signal. This criterion allows FASTTEST to ignore simple single-tone test signals, relatively-simple program material such as may be produced by a solo musical instrument, and conditions of silence.

2. Across all sections of the spectrum between tones in the reference signal, FASTTEST looks at the received signal to assure that its amplitude does not exceed a threshold of acceptability. This criterion allows FASTTEST to ignore complex voice and music program material which tends to have energy spread across much of the spectrum.

To permit user control of the triggering criteria, the allowable deviation from reference signal amplitude at generator tone frequencies (1 above) and the amount that energy at all other frequencies must be attenuated (2 above) are settable at three values. The Tight, Normal, and Loose selections each represents a different trade-off between the chance of false response on non-multitone signals versus the possibility of not triggering on legitimate multitone signals from a device with large amounts of noise and distortion and/or large deviations from flat frequency response. Select Tight for the minimum chance of false triggering This may be necessary when using very short generator waveform files (less than 2048 samples) since the consequent poorer frequency resolution makes it more difficult to discriminate between multitone signals and program material. Use Loose if FASTTEST will not otherwise trigger on highly distorted or noisy signals or signals passed through narrow-band or otherwise non-flat devices.

**Example** See example for `ATS2.Inst.FastTest.TransformLength`.



## Chapter 28

### FFT Spectrum Analyzer

---

#### ATS2.Inst.FFT.AcquireLength

Property

**Syntax**      `ATS2.Inst.FFT.AcquireLength`

**Data Type**    Constant

`apbFFTTrack`

Track FFT

`apbFFTAcq800`

800

`apbFFTAcq1500`

1.5k

`apbFFTAcq2500`

2.5k

`apbFFTAcq5k`

5k

`apbFFTAcq10k`

10k

`apbFFTAcq19k`

19k

`apbFFTAcq24k`

24k

`apbFFTAcq36k`

36k

`apbFFTAcq72k`

72k

*apbFFTAcq144k*  
144k

*apbFFTAcq256k*  
256k

**Description** This command sets the FFT Spectrum Analyzer acquire buffer record length.

**See Also** `ATS2.Inst.FFT.TransformLength`

**Example** See example for `ATS2.Inst.FFT.Averages`.

---

## ATS2.Inst.FFT.Averages

## Property

**Syntax** `ATS2.Inst.FFT.Averages`

**Data Type** Constant

<i>apbFftAvg1</i>	1
<i>apbFftAvg2</i>	2
<i>apbFftAvg4</i>	4
<i>apbFftAvg8</i>	8
<i>apbFftAvg16</i>	16
<i>apbFftAvg32</i>	32
<i>apbFftAvg64</i>	64
<i>apbFftAvg128</i>	128
<i>apbFftAvg256</i>	256

<i>apbFftAvg512</i>	512
<i>apbFftAvg1k</i>	1024
<i>apbFftAvg2k</i>	2048
<i>apbFftAvg4k</i>	4096

**Description** This command sets the FFT Spectrum Analyzer number of FFT averages.

FFT has the ability to average a number of successive acquisitions and spectrum analyses of a signal and display the averaged result. Since noise is random in amplitude and phase, averaging a succession of noise measurements results in a degree of cancellation and the averaged result will have less variance than the initial acquisition. Coherent signals, however, are the same at each acquisition and thus are not affected by averaging. Thus, spectral averaging will reduce the maximum peak excursions of the noise baseline in a typical signal spectrum while not affecting continuous signals, making it easier to detect and measure low amplitude signals and distortion products. Averaging over many seconds or minutes of program material such as music or voice may also be useful in order to determine the long-term average amplitude versus frequency distribution.

**See Also** `ATS2.Inst.FFT.AveragesType`

**Example**

```
Sub Main
  ATS.File.OpenTest "FFTTEST1.ATS2"
  With ATS2.Inst.FFT
    .InputFormat = apbFFTAnalog
    .AverageType = apbFFTPower
    .Averages = apbFFTAvg16
    .AcquireLength = apbFFTAcq10k
    .TransformLength = apbFFT8k
    .StartTime("sec") = 0
    .SubtractDC = apbFFTDSubAvg
    .WfmDisplay = apbFFTInterpolate
```

```

        .Window = apbFFTBlackmanHarris
        .TrigDelay("sec") = 0.000000
        .TrigSource = apbFFTOff
        .TrigSensitivity("dBFS") = -59.999594
        .TrigPolarity = apbPos
    End With
    ATS.Sweep.Start
End Sub

```

---

## ATS2.Inst.FFT.AveragesType

## Property

**Syntax**      **ATS2.Inst.FFT.AveragesType**

**Data Type**      Constant

The following list contains the selections relevant to the `ATS2.Inst.FFT.Window` command Blackman-Harris, Hann, Flat-Top, Equiripple, and None Window selections.

*apbFFTPower*  
Power (spectrum only)

*apbFFTSyncRealign*  
Sync, re-align

*apbFFTSync*  
Sync

The following list contains the selections relevant to the `ATS2.Inst.FFT.Window` command None, move to bin center Window selection.

*apbFFTPower*  
Power (spectrum only)

*apbFFTSyncRealignCenter*  
Sync, re-align, move center first

*apbFFTSyncRealignAverage*  
Sync, re-align, average first

*apbFFTSyncCenter*

Sync, move center first

*apbFFTSyncAverage*

Sync, average first

**Description** This command sets the type of Averaging the FFT Spectrum Analyzer uses when producing Time and Frequency domain measurements.

This command enables or disables computation of the average value of all samples in the acquisition buffer and subtraction that computed value from the value of each sample before an FFT transform or processing the values according to the Wave Display field and sending the results to the computer for display. The effect of the Subtract Average Value function is thus very similar having used AC coupling before acquiring the signal, as long as no signal peaks exceeded digital full scale. Use of the Subtract Average Value function may be valuable when examining low-level signals which contain a significant amount of DC offset, particularly in time domain (oscilloscope) presentations where the DC offset might otherwise cause the signal to be off-screen at the selected vertical span.

**See Also** `ATS2.Inst.FFT.Averages`

**Example** See example for `ATS2.Inst.FFT.Averages`.

---

## ATS2.Inst.FFT.Rdg

## Property

**Syntax** `ATS2.Inst.FFT.Rdg` (ByVal *Channel* As Constant, ByVal *Unit* As String)

**Data Type** Double

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits.

**Description** This command returns a unsettled reading for the FFT Spectrum Analyzer Peak Monitor meter and zeros the ready count.

**See Also** `ATS2.Inst.FFT.Ready`, `ATS2.Inst.FFT.Trig`

**Example**

```
Sub Main
    ATS.File.OpenTest "FFTTEST2.ATS2"
    With ATS2.Inst.FFT
        .Trig(apbChA)
    Do
        Ready = .Ready(apbChA)
    Loop Until .Ready(apbChA)
    AReading = ATS2.Inst.FFT.Rdg(apbChA, "FFS")
    ATS.Prompt.Text = "Ch A Monitor " _
        & Left(Str$(AReading),6) & "FFS"
    ATS.Prompt.ShowWithContinue
    Beep
    Stop
End Sub
```

---

## ATS2.Inst.FFT.Ready

## Property

**Syntax** `ATS2.Inst.FFT.Ready (ByVal Channel As Constant)`

**Data Type** Integer

<code>0</code>	Reading not ready.
<code>&gt;0</code>	Reading ready.

Parameter	Name	Description
	<code>Channel</code>	apbChA = Select channel A apbChB = Select channel B

**Description** This command returns the FFT Spectrum Analyzer Peak Monitor meter unsettled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any

number of times. Only a call to the `ATS2.Inst.FFT.Rdg` or `ATS2.Inst.FFT.Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.Inst.FFT.Rdg` command will be guaranteed to return quickly.

**See Also** `ATS2.Inst.FFT.Rdg`, `ATS2.Inst.FFT.Trig`

**Example** See example for `ATS2.Inst.FFT.Rdg`.

---

## ATS2.Inst.FFT.Trig

## Method

**Syntax** `ATS2.Inst.FFT.Trig(ByVal Channel As Constant)`

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Inst.FFTGen.Rdg` command. The reading in progress is aborted.

**See Also** `ATS2.Inst.FFT.Rdg`, `ATS2.Inst.FFT.Ready`

**Example** See example for `ATS2.Inst.FFT.Rdg`.

---

## ATS2.Inst.FFT.InputFormat

## Property

**Syntax** `ATS2.Inst.FFT.InputFormat`

Data Type	Constant
	<i>apbFftDigital</i> Digital
	<i>apbFftAnalog</i> Analog
	<i>apbFftJitterSec</i> ChA Analog / ChB Jitter(UI)

*apbFftJitterUI*

ChA Analog / ChB Jitter(sec)

**Description** This command sets the FFT Spectrum Analyzer Input Format.**Example** See example for `ATS2.Inst.FFT.Averages`.

---

**ATS2.Inst.FFT.StartTime****Property****Syntax** `ATS2.Inst.FFT.StartTime (ByVal Unit As String)`**Data Type** Double      The acceptable range of numbers depends upon the sample Rate set on the Digital I/O panel, since the acquisition buffer is a fixed length in samples. At a 48 kHz sample rate, for example, the Start Time field will accept numbers between plus and minus 341 milliseconds.

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: sec.

**Description** This command sets the FFT Spectrum Analyzer Start Time.

FFT permits the user to select any point in the acquired signal record as the beginning of the portion to be transformed. The FFT transform is then computed for the contiguous section of samples starting at that sample and continuing for the number of samples chosen in the Length field. FFT thus permits selective spectrum analysis of different sections of a complex signal such as program material or special test signals such as sinewave bursts.

If the original signal acquisition (F9) was made with a negative value in the Pre-trigger Time field, negative values up to and including that same value may be used as FFT Start Time values to permit spectrum analysis of the pre-trigger section of the acquired record.

**Example** See example for `ATS2.Inst.FFT.Averages`.



**ATS2.Inst.FFT.SubtractDC****Property**

<b>Syntax</b>	<code>ATS2 . Inst . FFT . SubtractDC</code>	
<b>Data Type</b>	Constant	
	<i>0</i>	DC Coupled
	<i>1</i>	Subtract Average
	<i>2</i>	Subtract 1/2pk-pk
<b>Description</b>	This command sets the FFT Spectrum Analyzer DC offset processing mode.	
<b>Example</b>	See example for <code>ATS2 . Inst . FFT . Averages</code> .	

**ATS2.Inst.FFT.TransformLength****Property**

<b>Syntax</b>	<code>ATS2 . Inst . FFT . TransformLength</code>	
<b>Data Type</b>	Constant	
	<i>apbFFT256</i>	256
	<i>apbFFT512</i>	512
	<i>apbFFT1k</i>	1024
	<i>apbFFT2k</i>	2048
	<i>apbFFT4k</i>	4096
	<i>apbFFT8k</i>	8192
	<i>apbFFT16k</i>	16384
	<i>apbFFT32k</i>	32768

**Description** This command sets the FFT Spectrum Analyzer FFT length.

**Example** See example for `ATS2.Inst.FFT.Averages`.

---

## ATS2.Inst.FFT.TrigDelay

## Property

**Syntax** `ATS2.Inst.FFT.TrigDelay (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: sec.

**Description** This command sets the FFT Spectrum Analyzer Trigger Delay time.

FFT has the ability to fill the acquisition buffer with signal samples starting at a user-defined time before the trigger occurs, then continuing until the buffer is full. This permits analysis of signal conditions both before and after the triggering event. A negative value entered in the Trigger Delay field determines how much time (and how many samples) prior to the trigger event are retained. The Pre-Trigger Time field is visible only on the large form of the Digital Analyzer panel. The total length of signal acquired will be as set in FFT Transform Length, with the remainder of the acquisition buffer filled after the trigger. For example, with maximum memory the length of the acquisition buffer for each channel is 341 milliseconds at a 48 kHz rate. If the Pre-Trigger Time value is -50 milliseconds, for example, then 291 additional milliseconds of signal following the trigger will also be acquired to fill the entire 341 ms buffer.

Pre-trigger data is acquired in this fashion: when the F9 key is pressed or Go is clicked, FFT and the DSP module immediately begin acquiring data samples, even though no trigger event may have yet occurred. If the acquisition buffer should completely fill before a trigger event occurs, data continues to be acquired in a FIFO (first in first out) basis with the oldest data being dropped as new data is added. When the trigger event occurs, FFT effectively creates a marker at that location (time zero) and another marker at the pre-trigger time before time zero and continues acquiring until every

location up to the pre-trigger marker is filled. Any portion from the pre-trigger time through time zero to the end of the record may then be displayed in oscilloscope fashion or transformed and viewed as a spectrum analysis.

**Example** See example for `ATS2.Inst.FFT.Averages`.

---

## ATS2.Inst.FFT.TrigLevel

## Property

**Syntax** `ATS2.Inst.FFT.TrigLevel (ByVal Unit As String)`

**Data Type** Double The acceptable range of numbers is between plus and minus 1 for the FFS unit.

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: FFS, FS, and dBFS.

**Description** This command sets the FFT Spectrum Analyzer Trigger Level. This control determines the signal level that must be obtained before a trigger event can occur. Once the Trigger event occurs the next and following samples are placed into the acquisition buffer until full.

**Example** See example for `ATS2.Inst.FFT.Averages`.

---

## ATS2.Inst.FFT.TrigPolarity

## Property

**Syntax** `ATS2.Inst.FFT.TrigPolarity`

**Data Type** Constant

*apbTrigSlopePos*  
Positive: time zero will be the first positive-going zero crossing of the trigger signal selected in the Trigger Source field.

*apbTrigSlopeNeg*  
Negative: time zero will be the first negative-going zero crossing of the selected trigger signal.

**Description** This command sets the FFT Spectrum Analyzer trigger polarity.

**Example** See example for `ATS2.Inst.FFT.Averages`.

---

## ATS2.Inst.FFT.TrigSensitivity Property

**Syntax** `ATS2.Inst.FFT.TrigSensitivity (ByVal Unit As String)`

**Data Type** Double      The acceptable range of numbers is between plus and minus 1 for the FFS unit.

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: FFS, FS, and dBFS.

**Description** This command sets the FFT Spectrum Analyzer Trigger Sensitivity. This control determines the signal level that must be obtained before a zero crossing trigger event can occur.

**Example** See example for `ATS2.Inst.FFT.Averages`.

---

## ATS2.Inst.FFT.TrigSource Property

**Syntax** `ATS2.Inst.FFT.TrigSource`

**Data Type** Constant

*apbFFTOff*  
Free Run: signal acquisition begins immediately after F9 or Go is initiated, regardless of signal amplitude. This is the typical operating mode with steady-state test signals.

*apbFFTChAAuto*  
Ch. A Auto:

*apbFFTChAFixed*  
Ch. A Fixed:

*apbFFTChBAuto*  
Ch. B Auto:

*apbFFTChBFixed*

Ch. B Fixed:

*apbFFTEExternal*

Trig In (Ext): The External selection refers to pin 3 of the 15-pin D-sub connector on the rear of the DSP module. This source is operational only with the SYS-2300 series Dual Domain units. If pin 3 is high (or open circuit, in which case it is pulled high by an internal pull-up resistor), triggering occurs at the next digital sample. Pulling pin 3 low from an external device holds off triggering, with acquisition being triggered on the next sample after pin 3 is pulled high. This External selection is unaffected by the Slope buttons.

*apbFFTDGen*

Digital Gen: The Digital Generator selection functions only on Dual Domain units (SYS-2300 series). If the Digital Generator is generating any of the waveforms selectable in the Waveform field, a Digital Generator trigger occurs at each zero crossing of the waveform, positive-going or negative-going as selected by the Slope buttons. If the Digital Generator is generating a signal from a waveform file, a Digital Generator trigger occurs as the first sample is read from the waveform file.

*apbFFTAGen*

Analog Gen: The Analog Generator Sync selection is the same signal as at the Generator Aux Signals Sync Output BNC on the front panel of ATS. This signal is a squarewave at the Analog Generator frequency in sinewave and squarewave waveforms, the envelope of the burst signal in all Burst waveforms, a squarewave at the lower IMD frequency in SMPTE IMD waveform, a squarewave at 1/2 the frequency spacing in CCIF IMD waveform, the squarewave IMD signal in DIM IMD waveform, and a pulse at the pseudo-random repetition rate in Pseudo noise modes. There is no signal in Random noise modes.

*apbFFTACMains*

Line (Mains): the power line frequency.

*apbFFTJitterGen*

Jitter Gen: The Digital Input/Output Jitter Generator selection provides a trigger at each positive or negative zero crossing for the selected waveform type.

*apbFFTChAFixedLev*

Ch. A Fixed Level

*apbFFTChBFixedLev*

Ch. B Fixed Level

**Description** This command sets the FFT Spectrum Analyzer Trigger Source.

The four channel A and channel B selections are software triggers, monitoring the signal (which may come from Digital or A/D sources) on the specified channel. channel A Fix and channel B Fix use a fixed threshold of 1.0%FS (-40 dBFS) on the channel referred to as the triggering threshold, and will trigger on the first signal excursion of the selected slope (Positive or Negative radio button) above that amplitude. The channel A and B Auto selections will cause triggering at one-half the peak-to-peak value if the selected channel has a signal amplitude greater than digital zero.

**Example** See example for `ATS2.Inst.FFT.Averages`.

---

## ATS2.Inst.FFT.WfmDisplay

## Property

**Syntax** `ATS2.Inst.FFT.WfmDisplay`

**Data Type** Constant

*apbFFTInterpolate*

Interpolate

*apbFFTSample*

Display Samples

*apbFFTPeak*

Peak Values

*apbFFTAbsolute*

Absolute Values

**Description** This command sets the FFT Spectrum Analyzer generator waveform display mode.

When Interpolate is selected, the DSP module will perform an interpolation calculation based on the assumption that the signal was band-limited by a low-pass filter before sampling. The Interpolate selection produces a much more accurate display of the signal waveform when the signal frequency is high (such as sample rate/100 or higher).

When Display Samples is selected, no processing takes place in the DSP module. At each time value plotted on the X-axis, the DSP simply sends the amplitude of the nearest-in-time acquired sample to the computer for plotting. When the signal frequency is low compared to the sample rate, this may produce an acceptable representation of the original signal waveform. At high signal frequencies, the waveform may be entirely unrecognizable in the Display Samples mode. For example, a 16 kHz sinewave acquired at the 48 kHz sample rate will have each cycle of waveform represented by only three amplitude samples and the result will look very little like a sinewave. The Display Samples mode may be useful when examining the true quantization-limited waveforms of very low amplitude digital domain signals.

When Peak Values is selected, the DSP searches all sample amplitudes in the acquisition buffer between each pair of X-axis time values plotted and returns to the computer the largest positive or negative value in that span, preserving the sign. The intended use of the Peak Values mode is when graphing a relatively long time span on the X-axis, where the combination of Start-to-Stop time span and Steps value on the Sweep panel results in skipping across many actual acquired samples between plotted points. For example, assume a signal is acquired at the 48 kHz sample rate (20.8 microseconds between samples). If the waveform of that signal is being viewed from 0 to 200 milliseconds with 400 steps, the time span between plotted points on the graph X-axis is 0.5 milliseconds (500 microseconds). There are approximately 24 samples between plotted points. If Peak Values or Absolute Values modes are not used, an unfortunate combination of signal frequency, X-axis span, and Points value can make it appear that no waveform, a near-DC signal,

or a waveform at a completely different frequency is present. Since Peak Values searches through all sample values within each span between plotted points and sends the largest value to be plotted, signals cannot be missed.

When Absolute Values mode is selected, the DSP searches all sample amplitudes in each plotted-point-to-plotted-point span as it does in Peak Values mode, but takes the absolute value of the largest positive or negative value and always sends a positive number to the computer. The advantage of Absolute Values mode is that logarithms may be computed when all numbers are positive, so a dB unit may be used on the Y axis to display the waveform. Waveform display with Absolute Values mode can create a wide dynamic range oscilloscope which displays the envelope of an audio signal, calibrated in familiar dB units such as dBV, dBu, etc. Absolute Values mode is most effective when the X-axis span and Points values are selected to produce approximately two plotted points per cycle of the waveform being plotted. For example, if an envelope display of tone burst waveforms of a 1 kHz signal (1 millisecond period) are being plotted across a 50 millisecond span, the Points value on the Sweep panel should be set to approximately 100.

**Example** See example for `ATS2.Inst.FFT.Averages`.

---

## ATS2.Inst.FFT.Window

## Property

<b>Syntax</b>	<code>ATS2.Inst.FFT.Window</code>
<b>Data Type</b>	Constant
	<code>apbFftBlackmanHarris</code> Blackman-Harris
	<code>apbFftHann</code> Hann
	<code>apbFftFlatTop</code> Flat-Top
	<code>apbFftEquiripple</code> Equiripple



<i>apbFftNone</i>	None
<i>apbFftNoneMoveToBinCenter</i>	None, move to bin center
<i>apbFftHamming</i>	Hamming
<i>apbFftGaussian</i>	Gaussian
<i>apbFftRifeVincent4</i>	Rife-Vincent 4
<i>apbFftRifeVincent5</i>	Rife-Vincent 5

**Description** This command sets the FFT Spectrum Analyzer Window selection. See Appendix C for FFT Window Descriptions.

**See Also** `ATS2.Inst.FFT.AveragesType`

**Example** See example for `ATS2.Inst.FFT.Averages`.

User Notes

### ATS2.Inst.Harmonic.AmplRdg

Property

**Syntax** `ATS2.Inst.Harmonic.AmplRdg` (ByVal *Channel* As Constant, ByVal *Unit* As String)

**Data Type** Double

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Unit</i>	The following units are available: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.

**Description** This command returns a settled reading for the Harmonic Analyzer Fundamental Amplitude meter and zeros the ready count.

**See Also** `ATS2.Inst.Harmonic.AmplReady`,  
`ATS2.Inst.Harmonic.AmplSettling`,  
`ATS2.Inst.Harmonic.AmplTrig`

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.Inst.Program = apbInstHarmonicAnalyzer
    With ATS2.Inst.Harmonic
        .InputFormat = apbInstInAnalog
        .AmplSettling(apbChA, 1.0, 0.000001, "v", _
            3, 0.03, apbExponential)
        .AmplSettling(apbChA, 0.5, 0.01, "Hz", 3, _
            0.03, apbExponential)
    End With
    Wait .5
    .AmplTrig(apbChA)
    .FreqTrig(apbChA)
End Sub
```

```

Do
Loop Until .AmplReady(apbChA) _
    And .FreqReady(apbChA)
var1 = .AmplRdg(apbChA, "V")
var2 = .FreqRdg(apbChA, "Hz")
End With
Text1$= "Channel A Fundamental Amplitude " & _
    Str$(Format(var1, "##.000")) & "V"
Text2$= "Channel A Fundamental Frequency " & _
    Str$(Format(var2, "##.000")) & "Hz"
ATS.Prompt.Text = Text1$ & vbCr & Text2$ _
ATS.Prompt.ShowWithContinue
Stop
End Sub

```

## ATS2.Inst.Harmonic.AmplReady

## Property

**Syntax** `ATS2.Inst.Harmonic.AmplReady (ByVal Channel As Constant)`

**Data Type** Integer

0 Reading not ready.  
>0 Reading ready.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** This command returns the Harmonic Distortion Analyzer Fundamental Amplitude settled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the `ATS2.Inst.Harmonic.AmplRdg` or `ATS2.Inst.Harmonic.AmplTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.Inst.Harmonic.AmplRdg` command will be guaranteed to return quickly.

**See Also** `ATS2.Inst.Harmonic.AmplRdg`,  
`ATS2.Inst.Harmonic.AmplSettling`,  
`ATS2.Inst.Harmonic.AmplTrig`

**Example** See example for `ATS2.Inst.Harmonic.AmplRdg`.

---

## ATS2.Inst.Harmonic.AmplSettling

## Method

**Syntax** `ATS2.Inst.Harmonic.AmplSettling` (ByVal *Channel* As Integer, ByVal *Tolerance* As Double, ByVal *Floor* As Double, ByVal *FloorUnit* As String, ByVal *Points* As Integer, ByVal *Delay* As Double, ByVal *Algorithm* As Constant)

**Parameter** See Appendix A for Settling Algorithm and parameter name descriptions.

**Description** This command sets the settling parameters for the `ATS2.Inst.Harmonic.AmplRdg` command.

**See Also** `ATS2.Inst.Harmonic.AmplRdg`,  
`ATS2.Inst.Harmonic.AmplReady`,  
`ATS2.Inst.Harmonic.AmplTrig`

**Example** See example for `ATS2.Inst.Harmonic.AmplRdg`.

---

## ATS2.Inst.Harmonic.AmplTrig

## Method

**Syntax** `ATS2.Inst.Harmonic.AmplTrig` (ByVal *Channel* As Constant)

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Inst.Harmonic.AmplRdg` command. The reading in progress is aborted.

**See Also** `ATS2.Inst.Harmonic.AmplRdg`,  
`ATS2.Inst.Harmonic.AmplReady`,  
`ATS2.Inst.Harmonic.AmplSettling`

**Example** See example for `ATS2.Inst.Harmonic.AmplRdg`.

---

## ATS2.Inst.Harmonic.Freq

## Property

**Syntax** `ATS2.AGen.Freq (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM

**Description** This command sets the Harmonic Distortion Analyzer tuning frequency.

**See Also** `ATS2.Inst.Harmonic.Tuning`

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.Inst.Selection = apbInstHarmonicAnalyzer
    With ATS2.Inst.Harmonic
        .Harmonics(apbChA, apbHarmSum1) = 2
        .Tuning = apbHarmFixed
        .Freq("Hz") = 1000.0
        .Selectivity = apbHarmHiSpeed
    End With
    ATS.Sweep.Data(1).Id = 6129
    ATS.Sweep.Data(1).Top("FFS") = 1.0
    ATS.Sweep.Data(1).Bottom("FFS") = 0.0
    ATS.Sweep.Source(1).Start("Hz") = 1000.0
```

```

ATS.Sweep.Source(1).Stop("Hz") = 3000.0
ATS.Sweep.Source(1).LogLin = apbLin
ATS.Sweep.Source(1).Steps = 100
ATS.Sweep.Start
ATS2.Inst.Harmonic.Selectivity = apbHarmHiAccuracy
ATS.Sweep.Append = True
ATS.Sweep.Start
ATS.Graph.Comment = "Hi-Speed and Hi-Accuracy _
    measurement bandwidth."
End Sub

```

---

## ATS2.Inst.Harmonic.FreqRdg

## Property

<b>Syntax</b>	<b>ATS2.Inst.Harmonic.FreqRdg</b> (ByVal <i>Channel</i> As Constant, ByVal <i>Unit</i> As String)	
<b>Data Type</b>	Double	
<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Unit</i>	The following units are available: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.
<b>Description</b>	This command returns a settled reading for the Harmonic Distortion Analyzer Frequency meter and zeros the ready count.	
<b>See Also</b>	ATS2.Inst.Harmonic.FreqReady, ATS2.Inst.Harmonic.FreqSettling, ATS2.Inst.Harmonic.FreqTrig	
<b>Example</b>	See example for ATS2.Inst.Harmonic.AmplRdg.	

---

## ATS2.Inst.Harmonic.FreqReady

## Property

<b>Syntax</b>	<b>ATS2.Inst.Harmonic.FreqReady</b> (ByVal <i>Channel</i> As Constant)
<b>Data Type</b>	Integer

0 Reading not ready.  
 >0 Reading ready.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
<b>Description</b>	<p>This command returns the Harmonic Distortion Analyzer Frequency settled reading ready count.</p> <p>Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the <code>ATS2.Inst.Harmonic.FreqRdg</code> or <code>ATS2.Inst.Harmonic.FreqTrig</code> commands will zero the ready count.</p> <p>If the reading is found to be ready, a call to the <code>ATS2.Inst.Harmonic.FreqRdg</code> command will be guaranteed to return quickly.</p>	
<b>See Also</b>	<code>ATS2.Inst.Harmonic.FreqRdg</code> , <code>ATS2.Inst.Harmonic.FreqSettling</code> , <code>ATS2.Inst.Harmonic.FreqTrig</code>	
<b>Example</b>	See example for <code>ATS2.Inst.Harmonic.AmplRdg</code> .	

## ATS2.Inst.Harmonic.FreqSettling

**Method**

<b>Syntax</b>	<b>ATS2.Inst.Harmonic.FreqSettling</b> (ByVal <i>Channel</i> As Integer, ByVal <i>Tolerance</i> As Double, ByVal <i>Floor</i> As Double, ByVal <i>FloorUnit</i> As String, ByVal <i>Points</i> As Integer, ByVal <i>Delay</i> As Double, ByVal <i>Algorithm</i> As Constant)
<b>Parameter</b>	See Appendix A for Settling Algorithm and parameter name descriptions.



**Description** This command sets the settling parameters for the `ATS2.Inst.Harmonic.FreqRdg` command.

**See Also** `ATS2.Inst.Harmonic.FreqRdg`,  
`ATS2.Inst.Harmonic.FreqReady`,  
`ATS2.Inst.Harmonic.FreqTrig`

**Example** See example for `ATS2.Inst.Harmonic.AmplRdg`.

---

## ATS2.Inst.Harmonic.FreqTrig

**Method**

**Syntax** `ATS2.Inst.Harmonic.FreqTrig`(ByVal *Channel* As Constant)

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Inst.Harmonic.FreqRdg` command. The reading in progress is aborted.

**See Also** `ATS2.Inst.Harmonic.FreqRdg`,  
`ATS2.Inst.Harmonic.FreqReady`,  
`ATS2.Inst.Harmonic.FreqSettling`

**Example** See example for `ATS2.Inst.Harmonic.AmplRdg`.

---

## ATS2.Inst.Harmonic.Harmonics

**Property**

**Syntax** `ATS2.Inst.Harmonic.Harmonics`(ByVal *Channel* As Constant, ByVal *Sum* As Constant)

**Data Type** Integer      Single decimal value or the sum of several decimal values representing multiple harmonics or a constant value as defined below.

Decimal value / Harmonic table:  
 2 = Harmonic 2 On

4 = Harmonic 3 On  
 8 = Harmonic 4 On  
 16 = Harmonic 5 On  
 32 = Harmonic 6 On  
 64 = Harmonic 7 On  
 128 = Harmonic 8 On  
 256 = Harmonic 9 On  
 512 = Harmonic 10 On  
 1024 = Harmonic 11 On  
 2048 = Harmonic 12 On  
 4096 = Harmonic 13 On  
 8192 = Harmonic 14 On  
 16384 = Harmonic 15 On

Constants table:

apbAll = Select all harmonics.

apbEven = Select all even harmonics.

apbNone = Select None of the harmonics.

apbOdd = Select all odd harmonics.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Sum</i>	apbSum1 = Select Sum 1 apbSum2 = Select Sum 2
<b>Description</b>	This command adds or removes individual or as a group harmonics from the Harmonic Distortion Analyzer RSS summation for the specified Channel and Sum number.	
<b>Example</b>	See example for <code>ATS2.Inst.Harmonic.Rdg</code> .	

## ATS2.Inst.Harmonic.InputFormat

## Property

<b>Syntax</b>	<code>ATS2.Inst.Harmonic.InputFormat</code>
<b>Data Type</b>	Constant

```

apbInstInDigital
    Digital
apbInstInAnalog
    Analog
    
```

**Description** This command sets the Harmonic Distortion Analyzer Input Format.

**Example** See example for `ATS2.Inst.Harmonic.FreqRdg`.

## ATS2.Inst.Harmonic.Rdg

## Property

**Syntax** `ATS2.Inst.Harmonic.Rdg (ByVal Channel As Constant, ByVal Sum As Integer, ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Sum</i>	apbSum1 = Select Sum 1 apbSum2 = Select Sum 2
	<i>Unit</i>	The following units are valid for the "Absolute Units" selection of the <code>ATS2.Inst.Harmonic.RelUnits</code> command: FFS, %FS, dBFS, Bits, V, Vp, Vpp, dBu, dBV, dBr 1, dBr 2.  The following units are valid for the "Relative Units" selection of the <code>ATS2.Inst.Harmonic.RelUnits</code> command: %, dB, PPM, X/Y

**Description** This command returns a settled reading for the Harmonic Distortion Analyzer Harmonic Sum meter and zeros the ready count.

**See Also** `ATS2.Inst.Harmonic.Ready`,  
`ATS2.Inst.Harmonic.Settling`,  
`ATS2.Inst.Harmonic.Trig`

**Example**

```

Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    
```

```

ATS2.Inst.Selection = apbInstHarmonicAnalyzer
With ATS2.Inst.Harmonic
    .InputFormat = apbInstInAnalog
    .Harmonics(apbChA, apbHarmSum1) = 2
    .Harmonics(apbChA, apbHarmSum2) = 4
    .RatioUnits(apbChA, apbHarmSum1) = False
    .RatioUnits(apbChA, apbHarmSum2) = False
    .Settling(apbChA, apbHarmSum1, 0.5, -1.2e+002, _
        "dBV", 1, 0.002, apbFlat)
    .Settling(apbChA, apbHarmSum2, 0.5, -1.2e+002, _
        "dBV", 1, 0.002, apbFlat)
    .Trig(apbChA, apbHarmSum1)
    .Trig(apbChA, apbHarmSum2)
Do
Loop Until .Ready(apbChA, apbHarmSum1) _
    And .Ready(apbChA, apbHarmSum2)
var1 = .Rdg(apbChA, apbHarmSum1, "dBV")
var2 = .Rdg(apbChA, apbHarmSum2, "dBV")
End With
Text1$= "Channel A Sum1 " & _
    Str$(Format(var1, "##.000")) & "dBV"
Text2$= "Channel A Sum2 " & _
    Str$(Format(var2, "##.000")) & "dBV"
ATS.Prompt.Text = Text1$ & Chr$(13) & Text2$
ATS.Prompt.ShowWithContinue
Stop
End Sub

```

---

## ATS2.Inst.Harmonic.Ready

## Property

**Syntax**      **ATS2.Inst.Harmonic.Ready** (ByVal *Channel* As Constant, ByVal *Sum* As Integer)

**Data Type**    Integer

0                      Reading not ready.  
>0                      Reading ready.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Sum</i>	apbSum1 = Select Sum 1 apbSum2 = Select Sum 2
<b>Description</b>	<p>This command returns the Harmonic Distortion Analyzer Harmonic Sum settled reading ready count.</p> <p>Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the <code>ATS2.Inst.Harmonic.Rdg</code> or <code>ATS2.Inst.Harmonic.Trig</code> commands will zero the ready count.</p> <p>If the reading is found to be ready, a call to the <code>ATS2.Inst.Harmonic.Rdg</code> command will be guaranteed to return quickly.</p>	
<b>See Also</b>	<p><code>ATS2.Inst.Harmonic.Rdg</code>,  <code>ATS2.Inst.Harmonic.Settling</code>,  <code>ATS2.Inst.Harmonic.Trig</code></p>	
<b>Example</b>	<p>See example for <code>ATS2.Inst.Harmonic.Rdg</code>.</p>	

## ATS2.Inst.Harmonic.RatioUnits

## Property

**Syntax** `ATS2.Inst.Harmonic.RatioUnits` (ByVal *Channel* As Constant, ByVal *Sum* As Integer)

**Data Type** Boolean

*True* Ratio Units.  
*False* Absolute Units.

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B

*Sum*                      `apbSum1 = Select Sum 1`  
                               `apbSum2 = Select Sum 2`

**Description**      This command selects the units that are available for the `ATS2.Inst.Harmonic.Rdg` command

**Example**            See example for `ATS2.Inst.Harmonic.Rdg`.

## ATS2.Inst.Harmonic.Selectivity

## Property

**Syntax**            `ATS2.Inst.Harmonic.Selectivity`

**Data Type**        Integer

*apbHarmHiSpeed*

Hi-Speed: In this mode measurement selectivity is reduced to improve measurement speed. Rapid readings of changing conditions are best done in this mode.

*apbHarmHiAccuracy*

Hi-Accuracy: In this mode measurement selectivity is increased to reduce the noise contribution to the measurement. Frequency and amplitude accuracy is improved.

**Description**      This command sets the Harmonic Distortion Analyzer measurement mode.

**Example**            See example for `ATS2.Inst.Harmonic.Freq`.

## ATS2.Inst.Harmonic.Settling

## Method

**Syntax**            `ATS2.Inst.Harmonic.Settling`(ByVal *Channel* As Constant, ByVal *Sum* As Constant, ByVal *Tolerance* As Double, ByVal *Floor* As Double, ByVal *FloorUnit* As String, ByVal *Points* As Integer, ByVal *Delay* As Double, ByVal *Algorithm* As Constant)

**Parameter**        See Appendix A for Settling Algorithm and parameter name descriptions.

**Description** This command sets the settling parameters for the `ATS2.Inst.Harmonic.Rdg` command.

**See Also** `ATS2.Inst.Harmonic.Rdg`,  
`ATS2.Inst.Harmonic.Ready`,  
`ATS2.Inst.Harmonic.Trig`

**Example** See example for `ATS2.Inst.Harmonic.Rdg`.

---

## ATS2.Inst.Harmonic.Trig

**Method**

**Syntax** `ATS2.Inst.Harmonic.Trig (ByVal Channel As Constant, ByVal Sum As Integer)`

Parameter	Name	Description
	<i>Channel</i>	apbChA = Select channel A apbChB = Select channel B
	<i>Sum</i>	apbSum1 = Select Sum 1 apbSum2 = Select Sum 2

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Inst.Harmonic.Ch1Sum1Rdg` command. The reading in progress is aborted.

**See Also** `ATS2.Inst.Harmonic.Rdg`,  
`ATS2.Inst.Harmonic.Ready`,  
`ATS2.Inst.Harmonic.Settling`

**Example** See example for `ATS2.Inst.Harmonic.Ch1Sum1Rdg`.

---

## ATS2.Inst.Harmonic.Tuning

**Property**

**Syntax** `ATS2.Inst.Harmonic.Tuning`

**Data Type** Constant  
*apbHarmAuto*

Counter Tuned: the frequency value measured by the Harmonic Distortion Analyzer Frequency counter is the filter steering source. This function would be selected when

making measurements from an external signal such as reproduction of a Compact Disc or digital audio tape or reception of a digital signal from a distant source.

*apbHarmSweepTrack*

Sweep Track: the filter tracks the frequency of whichever generator is selected in the Source 1 or Source 2 fields of the Sweep panel.

*apbHarmAGenTrack*

AGen Track: harmonic measurements track the frequency of the Analog Generator. This mode is useful for testing A/D converters driven from an analog output.

*apbHarmDGenTrack*

DGen Track: harmonic measurements track the frequency of the Digital Generator. This mode would normally be used when sweeping digital input- digital output devices with stimulus coming from the Digital Generator.

*apbHarmFixed*

Fixed: harmonic measurements are fixed, based on the frequency entered in the Freq field, unless the filter is being deliberately varied as part of a sweep test.

**Description** This command sets the Harmonic Distortion Analyzer Tuning Source.

**Example** See example for `ATS2.Inst.Harmonic.Freq`.



## Chapter 30

### INTERVU Digital Interface Analyzer

---

#### ATS2.Inst.Intervu.AcquisitionPosition

Property

**Syntax**            `ATS2.Inst.Intervu.AcquisitionPosition`

**Data Type**        Constant

*apbIntervuPreTrig*  
Pre - Trigger

*apbIntervuPostTrig*  
Post - Trigger.

**Description**     This command configures the Digital Interface Analyzer Data Acquisition to occur before or after the trigger.

**See Also**         `ATS2.Trigger.Source`

**Example**          See example for `ATS2.Trigger.Parity`.

---

#### ATS2.Inst.Intervu.Averages

Property

**Syntax**            `ATS2.Inst.Intervu.Averages`

**Data Type**        Constant

*apbIntervuAvg1*  
1

*apbIntervuAvg2*  
2

*apbIntervuAvg4*  
4

```
apbIntervuAvg8
    8
apbIntervuAvg16
    16
apbIntervuAvg32
    32
apbIntervuAvg64
    64
apbIntervuAvg128
    128
```

**Description** This command sets the Digital Interface Analyzer number of acquisition-and-processing cycles to average.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS.Application.PanelClose (apbAnalogGen)
    ATS.Application.PanelClose (apbAnalyzer)
    ATS2.DGen.OutputOn = True
    ATS2.Inst.Selection = apbInstIntervuAnalyzer
    ATS2.Inst.Intervu.Averages = apbIntervuAvg16
    ATS2.Inst.Intervu.Window = apbIntervuNone
    ATS.Sweep.Data(1).Id = 6055
    ATS.Sweep.Source(1).Id = 5613
    ATS.S2CDio.OutJitterType = apbDioOutJitterSine
    ATS.S2CDio.OutJitterAmpl("sec") = 100e-9
    ATS.S2CDio.InFormat = apbDioInGenMon
    ATS.Sweep.Start
    ATS.Graph.OptimizeLeft
End Sub
```

---

## ATS2.Inst.Intervu.JitterDetection

## Property

**Syntax**      `ATS2.Inst.Intervu.JitterDetection`

**Data Type**

Constant

*apbIntervueStableBits*

Stable Bits: derives the stable reference clock at 1/4 the actual cell (bit) rate, synchronized to the beginning transition of the preamble. The serial signal consists of 32 cells (bits) per subframe and two subframes (left and right channels) per frame. The frame rate is equal to the sample rate of the embedded audio. Thus, there are 64 cells (bits) in a complete frame and the cell rate is 1/64 the audio sample rate. The first four cells of each subframe are the preamble. The preamble always starts with a three UI (1 1/2 cell) wide pulse followed by sequences of one UI, two UI, and three UI pulses which are different among the three possible preambles. There is no cell transition time within the preamble which is common to all three preambles. The highest rate at which transitions can be guaranteed to occur regularly is at 1/4 the cell rate, which includes the beginning and end of each preamble but no transitions within the preamble. This rate is 16 times the audio sample rate, so the effective jitter measurement bandwidth is eight times the audio sample rate (384 kHz at a 48 kHz sample rate).

*apbIntervueAllBits*

All Bits: derives the stable reference clock at the actual cell (bit) rate. Since there are 64 cells per frame and the frame rate is the audio sample rate, the reference clock is at 64 times the sample rate and the effective jitter measurement bandwidth is 32 times the audio sample rate (1.536 MHz at a 48 kHz sample rate). Since the preamble of each sub-frame will not have transitions at every cell boundary due to its three-UI-wide pulses (violations of bi-phase coding), the DSP interpolates where transitions would have occurred if the preamble did not violate bi-phase coding.

*apbIntervuePreamble*

Preambles: the average rate of the trailing edge of the first three-UI-wide pulse in each preamble as the stable clock reference. Each actual transition at the trailing edge of the first three-UI-wide preamble pulse is then compared to that reference (average value) to obtain jitter values for display

as jitter waveform, histogram of jitter, or FFT spectrum analysis of jitter. The three-UI pulse in a preamble is the most robust portion of the digital interface signal, since it is least affected by reduced bandwidth in the cable or system. Therefore, jitter measurements made with the Preamble Jitter Detection selection tend to be measurements of the intrinsic jitter in the transmitting device clock and are relatively unaffected by data jitter caused by reduced bandwidth. Since this derived reference clock rate is low (twice the audio sample rate), the effective jitter measurement bandwidth equals the audio sample rate when Preamble is selected.

#### *apbIntervueSquareRising*

Squarewave Rising: In addition to measuring jitter on an AES/EBU or SPDIF/EIAJ serial digital input signal, INTERVU can also measure jitter on any 28 kHz-13 MHz squarewave connected to the BNC digital input connector. This feature permits direct measurement of clock jitter on A/D and D/A converters.

The Squarewave Rising selection measures jitter on rising edges of the Squarewave signal.

Jitter is a measurement of the time deviation of zero crossings of a waveform compared to a reference perfect clock of the same average frequency. For AES/EBU and SPDIF/EIAJ waveforms, System Two determines the average clock frequency by measuring the frame rate of the serial digital input signal with a frequency counter. This frame frequency extraction circuitry is not functional for a squarewave signal, so the DIO panel Sample Rate field is not useful with squarewave input. INTERVU determines the average clock frequency to the best of its ability from its acquired signal. Since the acquired signal duration is approximately four milliseconds, the resulting frequency measurement is limited in resolution. The result is that the initial time domain graph of jitter of a squarewave input clock, plotted across the approximately four millisecond record duration, may appear as a ramp. The desired jitter

signal is the deviation from this ramp. The Compute Linearity function is used to extract variations from an underlying systematic variation such as this ramp.

*apbIntervueSquareFalling*

Squarewave Falling: The Squarewave Falling selection measures jitter on falling edges of the Squarewave signal.

Note: See additional text in Squarewave Rising selection above.

**Description** This command determines at which transitions the clock timing is compared to the interface signal.

**Example** See example for `ATS2.Inst.Intervu.WfmDisplay`.

---

## ATS2.Inst.Intervu.TrigPolarity

## Property

**Syntax** `ATS2.Inst.Intervu.TrigPolarity`

**Data Type** Constant

*apbPos*

Positive: The data acquisition will begin on the first positive-going zero crossing of the signal selected in the Trigger Source field.

*apbNeg*

Negative: The data acquisition will begin on the first negative-going zero crossing of the selected signal selected in the Trigger Source field.

**Description** This command sets the Digital Interface Analyzer Trigger Polarity for the for the Trigger Source field Jitter Generator, External, Common Mode Signal, Interfering Noise, Digital Generator, and Input Zero Crossing selections.

**See Also** `ATS2.Trigger.Source`

**Example** See example for `ATS2.Trigger.Parity`.

---

## ATS2.Inst.Intervu.WfmDisplay

## Property

**Syntax**            `ATS2 . Inst . Intervu . WfmDisplay`

**Data Type**        Constant

*apbIntervueInterpolate*

Interpolate: the DSP module will perform an interpolation calculation based on the fact that the signal was band-limited by an internal 30 MHz low-pass filter before sampling.

*apbIntervueSamples*

Display Samples: no processing takes place in the DSP module. At each time value plotted on the X-axis, the DSP simply sends the amplitude of the nearest-in-time acquired sample of the digital interface waveform to the computer for plotting. When displaying only a few pulses of the digital interface waveform, this is typically the best mode to use.

*apbIntervuePeakValues*

Peak Values: the DSP searches all sample amplitudes in the acquisition buffer between each pair of horizontal axis time values plotted and sends to the computer for plotting the largest positive or negative value in that span, preserving the plus or minus sign. The intended use of the Peak Values mode is when graphing pulse width histograms or a relatively long time span on the X-axis, where the combination of Start-to-Stop time span and Steps value on the Sweep panel results in skipping across many actual acquired samples between plotted points. If Peak Values mode is not used, an unfortunate combination of signal, X-axis span, and Points value can make it appear that no waveform, a near-DC signal, or a waveform at a completely different frequency is present. Since Peak Values searches through all sample values within each span between plotted points and sends the largest value to be plotted, signals cannot be missed.

*apbIntervueEyePattern*

Eye Pattern: Following acquisition of the digital interface signal and extraction of an average clock signal from it, the worst-case (nearest to zero Volts) amplitude is determined for each time increment relative to the beginning of each data cell. These values are plotted when Upper Eye Opening and Lower Eye Opening are selected as Data parameters, resulting in a plot of the worst-case inside of the eye.

**Description** This command provides four modes for processing the amplitude-versus-time relationship of a sampled digital interface signal before displaying the waveform. These modes are applicable to digital storage oscilloscope operation (amplitude versus time graphs) and histograms, but have no effect on FFT spectrum analysis.

### Example

```
Sub Main
  ATS.File.OpenTest "INTERVU1.ATS2"
  With ATS2.Inst.Intervu
    .WfmDisplay = apbIntervuInterpolate
    .JitterDetection = apbIntervuStableBits
    ATS2.Trigger.Source = apbTrigXmitChBSub
    .Window = apbIntervuBlackmanHarris
  End With
  ATS.Sweep.Start
End Sub
```

---

## ATS2.Inst.Intervu.Window

## Property

**Syntax**      **ATS2.Inst.Intervu.Window**

**Data Type**    Constant

*apbIntervuBlackmanHarris*

Blackman-Harris

*apbIntervuHann*

Hann

*apbIntervuFlatTop*

Flat-Top

*apbIntervuEquiripple*

Equiripple

*apbIntervuNone*

None

**Description** This command sets the Digital Interface Analyzer Window selection. See Appendix C for FFT Window Discriptions.

**Example** See example for `ATS2.Inst.Intervu.WfmDisplay`.



# Chapter 31

## Analyzer Instrument Selection and Reference

### ATS2.Inst.RefdBm

Property

**Syntax** `ATS2.Inst.RefdBm (ByVal Unit As String)`

**Data Type** Double Reference value.

Parameters	Name	Description
	<i>Unit</i>	The following units are available: Ohms.

**Description** This command sets the Analyzer dBm impedance value. This value of circuit impedance is used as the "R" value in the equation  $V^2/R$  to compute power from the measured voltage (V), followed by decibel conversion.

**See Also** `ATS2.AnalogIn.Impedance`.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.Output (apbChB) = False
    ATS2.AGen.Impedance = apbAgenHighZ
    ATS2.AGen.RefdBm ("Ohms") = 600
    ATS2.AGen.Config = apbAGenUnBal
    ATS2.AGen.Ampl (apbChA, "dBm") = 0.0
    ATS2.AGen.OutputOn = True

    ATS2.AnalogIn.Source (apbChA) = apbAnalogInBNC_Unbal
    ATS2.AnalogIn.Impedance (apbChA) = apbInputHighZ

    AP.Inst.RefdBm ("Ohms") = 600
    Reference = AP.Inst.RefdBm ("Ohms")
    AP.Inst.Analyzer.FuncSettling (apbChA, 1.0, _
        .000002, "V", 4, .05, apbFlat)
    AP.Inst.Analyze.FuncTrig (apbChA)
Do
```

```

    Ready = AP.Inst.Analyze.FuncReady(apbChA)
    Loop Until Ready > 0
    Reading1 = AP.Inst.Analyze.FuncRdg(apbChA, "dBm")
    Debug.Print "Channel A Amplitude = _
    ";Format(Reading1, "#.0000");" dBm _
    (";Reference;" Ohms)"
End Sub

```

**Output** Channel A Amplitude = -.0404 dBm ( 600 Ohms)

**Comment** This example requires that a XLR cable be connected between the Analog Generator channel A output and the Analog Analyzer channel A input.

## ATS2.Inst.RefdBr

## Property

**Syntax** `ATS2.Inst.RefdBr` (ByVal *Channel* As Constant, ByVal *Unit* As String)

**Data Type** Double

### Parameter

Name	Description
<i>Channel</i>	apbChA = Select channel A reference. apbChB = Select channel B reference.
<i>Unit</i>	String that designates the desired unit.

The following units are valid for the "Digital" selection of the `ATS2.Inst.Analyzer.InputFormat` command: FFS, %FS, dbFS, Bits, V, Vp, Vpp, dBu, dBV

The following units are valid for the "Analog" selection of the `ATS2.Inst.Analyzer.InputFormat` command: V, dBV, dBu

**Description** This command sets a reference value for the dBr A or dBr B unit.

### Example

```

Sub Main
    ATS.File.OpenTest "Ref1.ats2"
    ATS2.Inst.RefVFS("V") = 2
    ATS2.Inst.RefFreq("Hz") = 2000
    ATS2.Inst.RefdBr(apbChA, "FFS") = .5

```

```

ATS2.Inst.RefdBr(apbChB, "FFS") = .75
Wait .5
With ATS2.Inst.Analyzer
    F_reading = .FreqRdg(apbChA, "%Hz")
    L_reading = .LevelRdg(apbChA, "dBrB")
    A_reading = .FuncRdg(apbChA, "dBrA")
    V_reading = .FuncRdg(apbChA, "V")
End With
a$= "Ch A Level Reading _
    "+Left(Str$(L_reading),6)+"dBr 2"
b$= "Ch A Freq Reading _
    "+Left(Str$(F_reading),6) +"%Hz"
c$= "Function Meter Reading _
    "+Left(Str$(A_reading),6)+"dBr 1"
d$= "Function Meter Reading _
    "+Left(Str$(V_reading),6)+"V/FS"
ATS.Prompt.Text = a$ & vbCr & b$ & vbCr & c$ _
    & vbCr & d$
ATS.Prompt.ShowWithContinue
Beep
Stop
End Sub

```

---

## ATS2.Inst.RefdBrAuto

## Method

**Syntax**      **ATS2.Inst.RefdBrAuto**

**Result**      Boolean

*True*              dBr reference set.  
*False*             dBr reference NOT set.

**Description**      This command sets the Analyzer dBr Reference field(s).

The following logic is used to determine which meter reading is written into which reference field:

If the Function meter units selected on the Audio Analyzer panel are not either dBrA or dBrB, then the Channel A Level meter reading is written into the dBrA Reference field and the Channel B Level meter reading is written into the dBrB Reference field.

If the Function meter units are either dBrA or dBrB and the corresponding Level meter is not set to a dBr unit, the Function meter measurement is written into the corresponding dBr Reference field and the other dBr Reference field takes its value from the Level meter on the corresponding channel.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
    ATS2.AnalogIn.Source(apbChB) = apbAnalogInGenMon
    ATS.Sweep.Data(1).Top("dBr A") = 5.000000e-002
    ATS.Sweep.Data(1).Bottom("dBr A") = -5.000000e-002
    ATS.Sweep.Data(2).Id = 6342
    ATS.Sweep.Data(2).Top("dBr B") = 5.000000e-002
    ATS.Sweep.Data(2).Bottom("dBr B") = -5.000000e-002
    ATS2.Inst.RefdBrauto
    Return = ATS2.Inst.RefdBrauto
    If Return = True Then Debug.Print "Reference Set"
    ATS.Sweep.Start
End Sub
```

**Output**

```
Reference Set
```

**ATS2.Inst.RefFreq****Property**

**Syntax**      **ATS2.Inst.RefFreq**(ByVal *Unit* As String)

**Data Type**      Double

<b>Parameter</b>	<b>Name</b>	<b>Description</b>
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: Hz

**Description**      This command sets the Frequency value for the relative frequency units (octaves, decades, %Hz, etc) of the Analyzer Frequency counter.

**Example**      See example for `ATS2.Inst.RefdBrauto`.

---

**ATS2.Inst.RefFreqAuto****Method**

**Syntax**            **ATS2.Inst.RefFreqAuto**

**Result**            Boolean

**Description**      This command sets the Analyzer dBr Reference field(s).

The following logic is used to determine which meter reading is written into the reference field:

The following logic is used to determine which meter reading is written into which reference field for System Two: If the Function meter units selected on the Analog Analyzer panel are not either dBrA or dBrB, then the Channel A Level meter reading is written into the dBrA Reference field and the Channel B Level meter reading is written into the dBrB Reference field.

If the Function meter units are either dBrA or dBrB and the corresponding Level meter is not set to a dBr unit, the Function meter measurement is written into the corresponding dBr Reference field and the other dBr Reference field takes its value from the Level meter on the corresponding channel.

**Example**

```
Sub Main
  ATS.Application.NewTest
  ATS2.AGen.OutputOn = True
  ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
  ATS2.AnalogIn.Source(apbChB) = apbAnalogInGenMon
  ATS.Sweep.Data(1).Top("dBr A") = 5.000000e-002
  ATS.Sweep.Data(1).Bottom("dBr A") = -5.000000e-002
  ATS.Sweep.Data(2).Id = 6342
  ATS.Sweep.Data(2).Top("dBr B") = 5.000000e-002
  ATS.Sweep.Data(2).Bottom("dBr B") = -5.000000e-002
  ATS2.Inst.RefdBrAuto
  Return = ATS2.Inst.RefdBrAuto
  If Return = True Then Debug.Print "Reference Set"
  ATS.Sweep.Start
End Sub
```

**ATS2.Inst.RefVFS****Property**

**Syntax** `ATS2.Inst.RefVFS (ByVal Unit As String)`

**Data Type** Double

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following units are valid for this command: V

**Description** This command sets the V/FS value is the analog-to-digital scaling value. When testing an external Analog to Digital converter (A/D), the value of analog input voltage which produces digital full scale output may be typed into this field. The Level Monitor or Reading meter units may then be selected as V, Vp, Vpp, dBu, dBV, dBr A, or dBr B to express the measured digital amplitude in terms of the analog input value to the converter.

**Example** See example for `ATS2.Inst.RefCh1dBr`.

**ATS2.Inst.RefWatts****Property**

**Syntax** `ATS2.Inst.RefWatts (ByVal Unit As String)`

**Data Type** Double Set Watts reference Impedance value.

Parameter	Name	Description
	<i>Unit</i>	The following units are available: Ohms only.

**Description** This command sets the value of Analog Analyzer Watts reference impedance. The known external load impedance should be entered, from which the software computes power from the measured voltage and the equation  $V^2 / R$  where R is the reference impedance.

**Example**

```
Sub Main
    ATS.Application.NewTest
    ATS2.AGen.Config = apbAGenUnBal
    ATS2.AGen.RefWatts("Ohms") = 8
    ATS2.AGen.Ampl(apbChA, "W") = .001
    ATS2.AGen.OutputOn = True
    ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
```

```

ATS2.Inst.RefWatts ("Ohms") = 8
ATS2.Inst.Analyzer.FuncSettling (apbChA, 1, _
    .000002, "V", 4, .05, apbExponential)
ATS2.Inst.Analyzer.FuncTrig (apbChA)
Do
    Ready = ATS2.Inst.Analyzer.FuncReady (apbChA)
Loop Until Ready > 0
Reading1 = ATS2.Inst.Analyzer.FuncRdg (apbChA, "W")
Debug.Print "Output Power = " & _
    Format (Reading1, "#.0000");" Watts"
ATS2.AGen.OutputOn = False
End Sub

```

---

## ATS2.Inst.Selection

## Property

**Syntax**            **ATS2.Inst.Selection**

**Data Type**        Constant

*apbInstAudioAnalyzer*

Audio Analyzer : usable for analog or digital domain input signals. Measures frequency, level, and one of a number of more sophisticated parameters (amplitude, 2-channel ratio, crosstalk, THD+N, selective amplitude, IMD, and phase) on both stereo channels simultaneously.

*apbInstFFTAnalyzer*

FFT Spectrum Analyzer : usable for analog or digital domain input signals. Provides general-purpose time domain (oscilloscope) display of waveforms or frequency domain (spectrum analyzer) display of signals, including the received jitter signal. Features include double precision transforms for better than 140 dB dynamic range, pre-trigger, a variety of selectable transform lengths up through 32k, acquisition memory up to 256k, the ability to position the start of the transformed section anywhere in the acquired record, both synchronous (time domain) and FFT spectrum (power-law) averaging, eight windowing functions, a frequency correction technique which adjusts single

sinewave signals to bin center so that no window is necessary, and several types of waveform processing for display.

*apbInstIntervuAnalyzer*

INTERVU Digital Interface Analyzer : analyzes the AES/EBU or consumer digital interface input signal of digital interface models via a 80 MHz sample rate A/D converter. Displays eye patterns, waveform display or spectrum analysis of the digital interface signal, waveform display or spectrum analysis of the recovered jitter signal, triggers on interface errors or on selected sections of the signal including received or transmitted preambles or received channel status blocks, measures jitter of the entire signal or selected sections such as preambles, and performs statistical analysis and histogram display of parameters including amplitude, pulse width, and jitter.

*apbInstFastTestAnalyzer*

FASTTEST Multitone Audio Analyzer : usable for analog or digital domain input signals. Provides time or frequency domain views of the signal. With multitone test signals, performs post-FFT processing to measure frequency response, total distortion and noise, noise in the presence of test signal, crosstalk, and generates psychoacoustic masking curves. Trigger modes include external and free-running, or triggering only upon receipt of the specific multitone signal matching the reference signal presently loaded into the digital generator. Variable trigger delay may be set to allow audio processors to settle. Frequency error correction compensates for multitone signals coming from other Audio Precision test instruments, played back from digital reproducers with different clock rates, or recorded and reproduced from analog recorders with speed errors up to 3%. FASTTEST also tests low-bit-rate perceptual coders with multitone signals by summing quantization noise and distortion in critical bands and comparing the results to an embedded psychoacoustic model of the frequency masking effect in humans.



*apbInstHarmonicAnalyzer*

Harmonic Distortion Analyzer : usable for analog or digital domain input signals. Permits flexible, highly selective measurement of the amplitude of user-specified harmonic orders. The user may choose to measure any individual harmonic through the 15th, or the sum of any arbitrary selection of harmonic distortion products from 2nd through 15th. Harmonic Distortion Analyzer may be used with either analog or digital domain signals. It is a two-channel program with four measurement meters per channel: a selective amplitude meter for the fundamental component of the signal, a frequency counter for the fundamental component, and two identical distortion product summing meters. Each of these distortion summing meters can be set to include any desired combination of harmonic distortion products (through the 15th) as long as each product is within the bandwidth limitations determined by the sample rate. Harmonic Distortion Analyzer can thus measure THD (Total Harmonic Distortion) without noise for any specified set of harmonic products. This analyzer effectively operates as a real-time program, even though it is internally based on FFT technology.

Both channels of Harmonic Distortion Analyzer may be set to measure one signal, such as a single-channel signal or one channel of a stereo signal. This configuration provides four Distortion summing meters. Each meter sends data to a different trace on the graph. This permits, for example, simultaneously plotting the fundamental signal amplitude and the 2nd, 3rd, 4th, and 5th harmonic amplitudes as five graph traces during a single frequency sweep.

**Description** This command selects the ATS Analyzer Instrument type.

**Example** Sub Main

```
ATS.Application.NewTest
ATS2.AGen.OutputOn = True
ATS.Application.PanelClose(apbAnalogGen)
ATS.Application.PanelClose(apbAnalyzer)
```

```
ATS2.Inst.Selection = apbInstFFTAnalyzer
ATS2.Inst.FFT.InputFormat = apbFFTAnalog
ATS2.Inst.FFT.TransformLength = apbFFT32k
ATS2.Inst.Anlr.AcquireLength = apbFFTTrack
ATS2.Inst.FFT.Window = apbFFTNoneMoveToBinCenter
ATS.Sweep.Source(1).Id = 5515
ATS.Sweep.Source(1).Start("Hz") = 100000
ATS.Sweep.Data(1).Id = 6024
ATS.Sweep.Data(1).Top("dBV") = 26.020600
ATS.Sweep.Start
ATS.Graph.OptimizeLeft
End Sub
```

### ATS2.Speaker.Mute

Property

**Syntax**            **ATS2 . Speaker . Mute**

**Data Type**        Boolean

*True*                Output Muted

*False*               Output On

**Description**     This command sets the speaker output to ON or Muted.

**Example**

```
Sub Main
  ATS.Application.NewTest
  ATS2.AGen.Wfm.Sine.Stereo
  ATS2.AGen.ChBTrackA = False
  ATS2.AGen.Freq(apbChB, "Hz") = 2000.0
  ATS2.AGen.OutputOn = True
  ATS2.AnalogIn.Source(apbChA) = apbAnalogInGenMon
  ATS2.AnalogIn.Source(apbChB) = apbAnalogInGenMon

  ATS.Application.PanelOpen(apbSpeaker)

  ATS2 . Speaker . Volume = 100
  ATS2 . Speaker . Source = apbSpeakerChA
  ATS2 . Speaker . Mute = False
  Wait 2
  ATS2 . Speaker . Source = apbSpeakerChB
  Wait 2
  ATS2 . Speaker . Mute = True
End Sub
```

**ATS2.Speaker.Source****Property****Syntax**      **ATS2 . Speaker . Source****Data Type**      Constant*apbSpeakerChA*

Channel A

*apbSpeakerChB*

Channel B

*apbSpeakerFuncA*

Function A

*apbSpeakerFuncB*

Function B

*apbSpeakerChAChB*

Channel A and Channel B (Sum)

*apbSpeakerFuncAFuncB*

Function A and Function B (Sum)

*apbSpeakerChAFuncA*

Channel A and Function A (Sum)

*apbSpeakerChBFuncB*

Channel B and Function B (Sum)

**Description**      This command selects the monitoring location(s) for the speaker and the headphone jack outputs.**See Also**      *ATS2 . Speaker . Mode***Example**      See example for *ATS2 . Speaker . Mute .***ATS2.Speaker.Volume****Property****Syntax**      **ATS2 . Speaker . Volume****Data Type**      Integer

**Description** This command selects the speaker volume and the headphone jack output signal level.

**Example** See example for `ATS2.Speaker.Mute`.

User Notes

### ATS2.SWR.In

Property

**Syntax** `ATS2.SWR.In (ByVal Channel As Constant)`

**Data Type** Long 0 - 192

Parameter	Name	Description
	<i>Channel</i>	apbSwrChA = Select channel A apbSwrChB = Select channel B apbSwrChAB = Select Channel A and Channel B

**Description** This command sets the channel connections of the Input switchers. Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

**See Also** `ATS2.SWR.Offset`

**Example** See example for `ATS2.SWR.Mode`.

### ATS2.SWR.InOut

Property

**Syntax** `ATS2.SWR.InOut (ByVal Channel As Constant)`

**Data Type** Long 0 - 192

Parameter	Name	Description
	<i>Channel</i>	apbSwrChA = Select channel A apbSwrChB = Select channel B apbSwrChAB = Select Channel A and Channel B

**Description** This command sets the channel connections of the Input and Output switchers. Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

**See Also**      `ATS2.SWR.Offset`

**Example**      See example for `ATS2.SWR.Mode`.

---

## ATS2.SWR.Mode

## Property

**Syntax**      `ATS2.SWR.Mode`

**Data Type**    Constant

*apbSwrBIndependentA*

B independent from A: when selected, channels A and B may be independently set to any channel number within their range. This is the normal mode for most operation.

*apbSwrBDriveAll*

B = All outputs driven, A = off: when selected, the switcher B common input is connected to all 12 outputs on each Output switcher module and the A common input is disconnected. Both the A and B output fields will be gray and unavailable for settings in this mode since all connections are defined by the mode itself. This mode enables connection of a single generator signal to all device inputs, which may be a requirement of a burn-in rack or a life test.

*apbSwrBDriveAllExceptA*

B = All outputs driven except # selected for A: when selected, the A common input connects to the channel number entered in the A output field and the B common input connects to the remaining 11 channels on that switcher and to all 12 channels of all other Output switchers connected. The purpose of this mode is for worst-case crosstalk measurements, so that all except one channels of a multi-track or multi-channel recorder or mixing console are driven while the output signal from the one un-driven channel is measured. This mode is normally used with a nested sweep with Source 2 on the Sweep panel set to scan channel A input and output through all possible device channels while Source 1 is commonly set for a frequency



sweep to measure selective crosstalk across the audio spectrum.

**Description** This command sets the switcher output configuration

**Example**

```
Sub Main
  Dim switch As Integer, msg As String
  ATS2.AGen.Ampl(apbChA, "V") = 1.0
  ATS2.AGen.OutputOn = True
  ATS2.Swr.Mode = apbSwrBIndependentA
  ATS2.Swr.Offset(apbOffsetOutput) = 1
  ATS2.Swr.Offset(apbOffsetChB) = 2
  For Switch = 1 To 6
    ATS2.Swr.In(apbSwrChA) = Switch
    'ATS2.Swr.In(apbSwrChB) = Switch
    'ATS2.Swr.In(apbSwrChAB) = Switch
    'ATS2.Swr.Out(apbSwrChA) = Switch
    'ATS2.Swr.Out(apbSwrChB) = Switch
    'ATS2.Swr.Out(apbSwrChAB) = Switch
    'ATS2.Swr.InOut(apbSwrChA) = Switch
    'ATS2.Swr.InOut(apbSwrChB) = Switch
    'ATS2.Swr.InOut(apbSwrChAB) = Switch
    ATS2.Inst.Analyzer.LevelTrig(apbChA)
    While ATS2.Inst.Analyzer.LevelReady(apbChA) = 0
      Wend
    rdg = ATS2.Inst.Analyzer.LevelRdg(apbChA, "V")
    If rdg > 0.5 * signal Then
      msg = msg & "Ch A In " & Switch & _
        "<-> Ch A Out " & Switch + _
        ATS2.Swr.Offset(apbOffsetOutput) & Chr(13)
    End If
  Next Switch
  ATS.Prompt.FontSize = 10
  ATS.Prompt.Text = msg
  ATS.Prompt.ShowWithContinue
  Stop
End Sub
```

## ATS2.SWR.Offset

## Property

**Syntax** `ATS2.SWR.Offset (ByVal Type As Constant)`

**Data Type** Long 0 - 192

Parameter	Name	Description
	<i>Type</i>	apbOffsetOutput = Output offset relative to input of channel A or channel B. apbOffsetChB = Channel B offset relative to channel A.

**Description** This command determines the channel number difference between the input and output switcher channels and the channel A and channel B switchers.

**See Also** `ATS2.SWR.InOut`

**Example** See example for `ATS2.SWR.Mode`.

## ATS2.SWR.Out

## Property

**Syntax** `ATS.SWR2.Out (ByVal Channel As Constant)`

**Data Type** Long 0 - 192

Parameter	Name	Description
	<i>Channel</i>	apbSwrChA = Select channel A apbSwrChB = Select channel B apbSwrChAB = Select Channel A and Channel B

**Description** This command sets the channel connections of the Output switchers. Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

**See Also** `ATS2.SWR.Offset`

**Example** See example for `ATS2.SWR.Mode`.

## Chapter 34

### Sync/Ref Input

---

#### ATS2.Sync.FrameLock

Property

**Syntax**            `ATS2.Sync.FrameLock`

**Data Type**        Boolean

<i>True</i>	Enable
<i>False</i>	Disable

**Description**     This command synchronizes the output AES/EBU frame (status bits preambles) to the AES/EBU Ref Input frame (status bits preambles).  
  
The Digital Output Sample Rate (OSR) command `ATS2.Dio.OutRate` and Sync Reference Frequency command `ATS2.Sync.Freq` must be set identical for this control to operate correctly.

**See Also**            `ATS2.Sync.OutRate`

---

#### ATS2.Sync.Freq

Property

**Syntax**            `ATS2.Sync.Freq` (ByVal *Unit* As String)

**Data Type**        Double            8kHz - 54kHz

Parameter	Name	Description
	<i>Unit</i>	String that designates the desired unit. The following unit is valid for this command: Hz

**Description**     This command specifies the exact Sync Input rate to be assumed by the phase-locked loop which locks the internal crystal oscillator to the reference. The Internal Sample Rate is then derived from the internal crystal oscillator. Normally, the user will enter the known reference

frequency. If the value entered differs by small amounts (less than 15 ppm) from the actual Reference frequency, all ATS sample rates will be shifted by the percentage error. If the value entered differs by more than +/-15 ppm from the actual Reference signal frequency, the internal crystal oscillator will not lock to the reference. When either of the two video sync functions (NTSC or PAL/SECAM) is selected, the normal horizontal sync rate for the selected video standard is automatically typed into the Frequency field.

**See Also**

ATS2.Sync.

**Example**

```
Sub Main
    ATS2.Sync.Source = apbSyncNTSCHorzRate
    ATS2.Sync.Impedance = apbSync75ohm
    ATS2.Sync.Source = False
    rdg = ATS2.Sync.Freq("Hz")
    If (rdg < lower_limit) Or (rdg > upper_limit) Then
        'input sync freq not close enough, flag an _
        error and ...
    End
    Else
        If ATS2.Sync.OutRangeRdg Then
            'internal clock not sync'ed, flag an _
            error and ...
        End
    Else
        ATS2.Sync.Source = True
        'now perform further testing on DUT...
    End If
End If
End Sub
```

**ATS2.Sync.FreqRdg****Property**

**Syntax**      **ATS2.Sync.FreqRdg** (ByVal *Unit* As String)

**Data Type**      Double

**Parameter**

Name	Description
<i>Unit</i>	The following units are available, Hz.

**Description** This command returns a unsettled reading for the Sync Delay Input Frequency for the signal selected in the Sync Source field when the ON/OFF button is OFF. This is intended as a verification of a proper sync input connection. The Reference frequency value is usually known to a greater accuracy than it can be measured by ATS (whose accuracy is typically about 1 ppm), in which case the known value should be entered in the Sync Input Frequency entry field. For example, a measured and displayed value of 47.9998 kHz almost certainly indicates an actual 48 kHz reference frequency, and 48.0000 kHz is the value which should be entered in the Input Frequency entry field. When the ON/OFF button is turned ON, the display field is blanked since the reading will be identical to the value in the Frequency Entry field.

**See Also** `ATS2.Sync.FreqReady`, `ATS2.Sync.FreqTrig`

**Example**

```
Sub Main
  ATS2.Sync.Source = False
  ATS2.Sync.FreqTrig
  While ATS2.Sync.FreqReady = False
    'Do other tasks while waiting for reading ...
  Wend
  reading1 = ATS2.Sync.FreqRdg ("Hz")
  Debug.Print "Sync Input Frequency = " & _
    Format(Reading1, "#.0000") & " Hz"
End Sub
```

**Output** Sync Input Frequency = 48000.0017 Hz

---

## ATS2.Sync.FreqReady

## Property

**Syntax** `ATS2.Sync.FreqReady`

**Data Type** Integer

0 Reading not ready.  
>0 Reading ready.

**Description** This command returns the Sync Frequency unsettled reading ready count.

Because readings do not return until a reading is ready, this command may be used to avoid waiting for a reading. This command does NOT zero the ready count and so may be called any number of times. Only a call to the `ATS2.Sync.FreqRdg` command will zero the ready count.

If the reading is found to be ready, a call to the `ATS2.Sync.FreqRdg` command will be guaranteed to return quickly.

Note that readings free run at the selected measurement rate and eventually become ready without a call to the `ATS2.Sync.FreqTrig` command.

**See Also** `ATS2.Sync.FreqRdg`, `ATS2.Sync.FreqTrig`

**Example** See example for `ATS2.Sync.FreqRdg`.

---

## ATS2.Sync.FreqTrig

**Method**

**Syntax** `ATS2.Sync.FreqTrig`

**Description** Causes a restart of the reading cycle and zeros the ready count for the `ATS2.Sync.FreqRdg` command. The reading in progress is aborted.

**See Also** `ATS2.Sync.FreqRdg`, `ATS2.Sync.FreqReady`

**Example** See example for `ATS2.Sync.FreqRdg`.

---

## ATS2.Sync.Impedance

**Property**

**Syntax** `ATS2.Sync.Impedance`

**Data Type** Constant

*apbSyncHiZ*

Hi Impedance

*apbSync75ohm*

75 Ohms

<b>Description</b>	This command controls the input impedance for the Un-Balanced Sync Input.
<b>See Also</b>	<code>ATS2.Sync.Source</code>
<b>Example</b>	See example for <code>ATS2.Sync.Freq</code> .

---

## ATS2.Sync.OutOfRangeRdg

### Property

<b>Syntax</b>	<code>ATS2.Sync.OutOfRangeRdg</code>
<b>Result</b>	Boolean
	<i>True</i> In Range
	<i>False</i> Out Of Range
<b>Description</b>	This command returns a unsettled reading for the Sync Out Of Range indicator.
<b>Example</b>	See example for <code>ATS2.Sync.Freq</code> .

---

## ATS2.Sync.Source

### Property

<b>Syntax</b>	<code>ATS2.Sync.Source</code>
<b>Data Type</b>	Boolean
	<i>True</i> Enable.
	<i>False</i> Disable.
<b>Description</b>	This command enables or disables the external sync input.
<b>See Also</b>	<code>ATS2.Sync.SourceInput</code>
<b>Example</b>	See example for <code>ATS2.Sync.FreqRdg</code> .

---

## ATS2.Sync.SourceInput

### Property

<b>Syntax</b>	<code>ATS2.Sync.SourceInput</code>
<b>Data Type</b>	Constant

*apbSyncAESSyncRate*

AES Sync Rate:

*apbSyncSquarewave*

Squarewave:

*apbSyncNTSCHorzRate*

NTSC Video Sync Horz Rate:

*apbSyncPALHorzRate*

PAL / SECAM Video Sync Horz Rate:

**Description** This command sets the input type for the external sync input.

**See Also** `ATS2.Sync.Source`

**Example** See example for `ATS2.Sync.Freq.`



---

### ATS2.Trigger.Coding

Property

**Syntax**            `ATS2.Trigger.Coding`

**Data Type**        Boolean

*True*                Enable Data Acquisition trigger generation on a Coding error.  
*False*               Disable Data Acquisition trigger generation on a Coding error.

**Description**     This command enables or disables generation of a Data Acquisition trigger for a Coding Error.

Coding indicates a deviation from proper biphasic coding in the input serial stream (ignoring preambles). Proper biphasic signals can never remain at a logic high or logic low level for more than two consecutive Unit Intervals (UI) except in the preamble. The preamble deliberately deviates from biphasic coding in order to provide a unique frame synchronization signal, so preambles are excluded from the function of the Coding indicators.

**Example**            See example for `ATS2.Trigger.Parity`.

---

### ATS2.Trigger.Confidence

Property

**Syntax**            `ATS2.Trigger.Confidence`

**Data Type**        Boolean

*True*                Enable Data Acquisition trigger generation on a Confidence error.  
*False*               Disable Data Acquisition trigger generation on a Confidence error.

**Description** This command enables or disables generation of a Data Acquisition trigger for a Confidence Error.

The Confidence error occurs when the ratio between the amplitude of the three UI long pulse and the following one UI-long pulse in a preamble becomes large enough to cause an increasing probability of errors when slicing the received signal into logic high and low values. This large ratio occurs when the transmission bandwidth has been reduced to marginal or unacceptable values. Under these conditions, selection of hardware input equalization (XLR with EQ or BNC with EQ rather than XLR or BNC selections of the Input Format field) will often compensate for the cable bandwidth reduction, and provide reliable measurements.

**Example** See example for `ATS2.Trigger.Parity`.

---

## ATS2.Trigger.Lock

## Property

**Syntax** `ATS2.Trigger.Lock`

**Data Type** Boolean

*True* Enable Data Acquisition trigger generation on a Lock error.  
*False* Disable Data Acquisition trigger generation on a Lock error.

**Description** This command enables or disables generation of a Data Acquisition trigger for a Lock Error.

The Lock error occurs when the digital input phase-locked loop is unable to lock to the incoming signal.

**Example** See example for `ATS2.Trigger.Parity`.

---

## ATS2.Trigger.Parity

## Property

**Syntax** `ATS2.Trigger.Parity`

**Data Type** Boolean

*True* Enable Data Acquisition trigger generation on a Parity error.  
*False* Disable Data Acquisition trigger generation on a Parity error.

**Description** This command enables or disables generation of a Data Acquisition trigger for a Parity Error.

The Parity error indicates a parity error in either subframe. Correct parity is determined by comparing the P (parity) bit with the sum of the remaining 31 bits in each subframe. Any single bit error or odd number of bit errors introduced in transmission within a subframe will cause a Parity error indication, but even numbers of bit errors cannot be detected by this technique.

### Example

```
Sub Main
    ATS.Application.NewTest
    ATS.Application.PanelClose (apbAnalogGen)
    ATS2.Inst.Selection = apbInstIntervuAnalyzer
    ATS2.Trigger.Source = apbTrigRcvError
    ATS2.Inst.Intervu.TrigPolarity = apbPos
    ATS2.Inst.Intervu.AcquisitionPosition = _
        apbIntervuPreTrig
    ATS2.Trigger.Confidence = False
    ATS2.Trigger.Coding = False
    ATS2.Trigger.Lock = False
    ATS2.Trigger.Parity = True

    ATS2.Dio.InFormat = apbDioInGenMon
    ATS.Sweep.Data(1).Id = 6053
    ATS.Sweep.Source(1).Id = 5612
    ATS.Sweep.Source(1).Start("sec") = -5.0e-006
    ATS.Sweep.Source(1).Stop("sec") = 5.0e-006

    ATS.Application.SetWatchDogTimer1(5.0,False)
    ATS.Sweep.StartNoWait
    ATS.Graph.Comment = "Wait for Parity Error to
occur."

' The Sweep will proceed automatically when the
'   Parity Error occurs.

Do
    Loop While ATS2.Dio.FlagParityRdg = False
'Wait here for Parity Error to be detected on the _
    Digital Input/Output panel.
```

```

ATS.Graph.Comment = "Parity Error detected and _
                    waveform display updated."
ATS.Graph.OptimizeLeft

End Sub
Sub ATSEvent_OnWatchDogTimeout (ByVal Id As Long)
    If Id = 1 Then
        ATS2.Dio.OutParityError = True
    End If
End Sub

```

## ATS2.Trigger.Source

## Property

**Syntax**            **ATS2.Trigger.Source**

**Data Type**        Integer

- |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | Ch. A Receive Preamble: cause signal to be acquired at the first Channel A (left) Preamble which occurs after Go is clicked or the F9 function key is pressed. The Channel A Preamble is known as the X Preamble in the AES/EBU standard and the M Preamble in the Consumer standard. The first information acquired will be the last four Unit Intervals of the selected preamble, followed by the LSB of the audio signal if full 24-bit resolution audio is transmitted, or the beginning of the 4-bit Auxiliary data if audio is restricted to 20 bits or less. |
| 1 | Ch. A Transmit Preamble: cause signal to be acquired beginning at the start of the first Channel A Preamble which is transmitted from System Two after the <code>ATS.Sweep.Start</code> command is executed. The first information acquired includes the entire preamble, followed by audio or Auxiliary data. This triggering selection permits measurement of time delay through a digital device or system under test.                                                                                                                                           |
| 2 | Ch. B Receive Preamble: cause signal to be acquired at the first Channel B (right) Preamble which occurs after Go is clicked or the F9 function key is pressed. The Channel B                                                                                                                                                                                                                                                                                                                                                                                       |

- 3 Preamble is known as the Y Preamble (AES/EBU) or W Preamble (consumer). The first information acquired will be the last four Unit Intervals of the selected preamble, followed by the LSB of the audio signal if full 24-bit resolution audio is transmitted, or the beginning of the 4-bit Auxiliary data if audio is restricted to 20 bits or less.  
Ch. B Transmit Preamble: cause signal to be acquired beginning at the start of the first Channel B Preamble which is transmitted from System Two after `ATS.Sweep.Start` command is executed. The first information acquired includes the entire preamble, followed by audio or Auxiliary data. This triggering selection permits measurement of time delay through a digital device or system under test.
- 4 Receive Error: selection is a pre-trigger, causing the 256k samples (about 3.9 milliseconds) immediately preceding an interface Error Flag to be retained (approximately 39 microseconds of signal following the occurrence of the error will also be retained. The interface Error Flags are generated by the AES/EBU receiver chip of the DIO, and their status is indicated by the Parity, Coding, Lock, or Confidence indicators at the right of the DIO panel. If this acquisition trigger selection is in use and a Parity error, Coding error, Lock error, or Confidence error occurs, the last (approximately) 3.9 milliseconds of interface signal preceding the error will be retained in the INTERVU buffer for examination via waveform display, spectrum analysis, or probability histograms. The Invalid indicator is not considered an interface error and thus will not result in an acquisition into INTERVU.
- 5 Receive Block: causes signal to be acquired beginning at the end of the first Channel Status Block Preamble received after Go is clicked or the F9 function key is pressed. This is known as the Z Preamble in the AES/EBU standard and the B Preamble in the Consumer standard. The first information displayed will be the last four UIs of the Z preamble, followed by the LSB of the Channel A audio signal if full 24-bit resolution audio is transmitted, or the beginning of the 4-bit Auxiliary data if audio is restricted to 20 bits or less, of

	the frame which marks the beginning of a new Channel Status Block. Channel Status Blocks are 192 frames long, with the C (Channel Status) bit of each of these 192 frames being assembled into the 24 Channel Status Bytes defined in the AES/EBU and Consumer standards.
6	Jitter Generator: causes a trigger at every zero crossing of the sinewave, squarewave, or noise signal generated by the DIO jitter generator. This selection provides a stable display of the received jitter waveform when measuring jitter gain or loss through a digital device.
7	External:
8	Common Mode Signal:
9	Interfering Noise:
10	Digital Generator:
11	Transmit Block:
12	Ch.A Sync Preamble:
13	Ch.B Sync Preamble:
14	Sync Error:
15	Sync Block:
16	Input ZeroCrossing:

**Description** This command defines the trigger source that is used to trigger an acquisition.

**Example** See example for `ATS2.Inst.Intervu.WfmDisplay`.

---

## ATS2.Trigger.Source

## Property

<b>Syntax</b>	<code>ATS2.Trigger.Source</code>
<b>Data Type</b>	Constant
	<code>apbTrigAgen</code> Analog Generator
	<code>apbTrigDgen</code> Digital Generator

<i>apbTrigJitGen</i>	Jitter Generator
<i>apbTrigExt</i>	External Trigger In
<i>apbTrigLine</i>	Line (AC Mains)
<i>apbTrigRcvChAPreamJit</i>	ChA Rec Sub-Frame
<i>apbTrigRcvChBPreamJit</i>	ChB Rec Sub-Frame
<i>apbTrigRcvChAPreamDeJit</i>	ChA Rec Sub-Frame DeJitt
<i>apbTrigRcvChBPreamDeJit</i>	ChB Rec Sub-Frame DeJitt
<i>apbTrigRcvBlock</i>	Rec Block (192 frames)
<i>apbTrigRcvError</i>	Rec Error
<i>apbTrigXmitChAPreamJit</i>	ChA Xmit Sub-Frame
<i>apbTrigXmitChBPreamJit</i>	ChB Xmit Sub-Frame
<i>apbTrigXmitChAPreamDeJit</i>	ChA Xmit Sub-Frame DeJitt
<i>apbTrigXmitChBPreamDeJit</i>	ChV Xmit Sub-Frame DeJitt
<i>apbTrigXmitBlock</i>	Xmit Block (192 frames)
<i>apbTrigSyncChAPream</i>	ChA Sync/Ref Rec Sub-Frame
<i>apbTrigSyncChBPream</i>	ChB Sync/Ref Rec Sub-Frame

*apbTrigSyncBlock*

Sync/Ref Rec Block (192 frames)

*apbTrigSyncError*

Sync/Ref Error

**Description** This command defines the trigger source that is used to trigger an acquisition.

**Example** See example for `ATS2.Inst.Intervu.WfmDisplay`.



---

# Appendix A

## Settling Algorithm

---

### Description

The general concept of the Sweep Settling Exponential and Flat algorithms is to discard all meter readings during the Delay interval, then to compare the number of successive readings equal to the Points value against the Tolerance or Floor values. Only when the specified (Points) number of consecutive readings agree with one another within the specified Tolerance or Floor values will the data be considered settled. It is then accepted for plotting and the Sweep Source parameter permitted to proceed to the next step.

---

### Settling Parameter Descriptions

Name	Description
<i>Channel</i>	This parameter is not used by all settling commands. Review the Settling command syntax to determine if this parameter is required. apbChA = Select channel A apbChB = Select channel B
<i>FloorUnit</i>	String that designates the desired unit to be used with the Floor# Parameter. Refer to the reading to determine the appropriate unit selections.
<i>Tolerance</i>	The Tolerance value which should be entered is the amount of variability the user is willing to accept from test to test. A Tolerance value of 0.1% (about 0.01 dB) or even slightly smaller may be appropriate when making frequency response measurements on the test system itself or on an

external device known to be very flat and being measured under excellent signal-to-noise conditions. At the other extreme, Tolerance values of 10% to 25% (1 to 2 dB) may be required to obtain data under noisy conditions, or when making measurements with a random noise signal as the stimulus. The default value of 1% (about 0.1 dB) is a good starting compromise for most level measurements.

*Floor*

The Floor value is used by the algorithms instead of the Tolerance value whenever the Floor value is larger. When the measurements values are greater than a few percent of full scale on the instrument range in use, the Tolerance value is normally the determining parameter. If the measurements are very near the bottom of the instruments dynamic range, use of only a Tolerance parameter could result in a hang up situation, since the percentage difference between two adjacent values (quantization levels) at the bottom of a meters range is large. The Floor parameter thus serves as a safety valve, avoiding slowing or hang ups in the highly resolution-limited situations where the signal is near the bottom of a measurable range. The default values of Floor for each meter are chosen to be approximately the resolution of that meter on its most sensitive range. Since resolution varies with reading rate (slower reading rates give more resolution), it may be appropriate to change the default values when reading rate is fixed at a given value.

*FloorUnit*

String that designates the desired unit to be used with the Floor# Parameter. Refer to the reading to determine the appropriate unit selections.

*Points*

The value determines how many consecutive readings are examined by the Settling Algorithm to qualify a measurement to be returned for display.

*Delay*

The value determines how long ATS software waits at each new step of a sweep before starting to examine

measurements from the instrument. The Delay value is effective even when the Algorithm selection is None. The Delay time will be taken at the beginning of each nest of a nested sweep, including nested FFT measurements with the FFT at Source 1 and another parameter such as generator amplitude at Source 2. Acquisition of signal into any of the FFT programs will not begin until the Delay value (or 200 milliseconds, whichever is greater) has passed. For Time sweeps where it is desired to make as many measurements per second as possible, the Delay value should be set to zero in addition to selecting None for settling.

#### *Algorithm*

apbNone = None: no settling process takes place for that meter. However, the Delay value (see the Delay topic) is still implemented before each point is plotted even with None selected as the settling algorithm. Measurements such as wow and flutter, phase jitter, and (with ATS Dual Domain) interface signal jitter are examples of cases where no settling should be used, since it is normally desired to see the extreme variations in measurements.

apbExponential = Exponential: the newest reading (N) must agree with the immediately preceding reading (N-1) within the Tolerance value, with the reading before (N-2) that within twice the Tolerance value, with the reading before that (N-3) within four times the Tolerance value, etc. Exponential is the recommended settling algorithm for most audio applications, since typical device transients tend to die away in an exponential fashion. Exponential thus will usually provide repeatable results to the Tolerance acceptable to the user in the minimum length of time.

apbFlat = Flat: the percentage difference between each set of two consecutive readings (N vs N-1, N-1 vs N-2, etc.) must be equal to or less than the specified Tolerance value, through the number of readings specified as the Points value. Illustrating the Flat algorithm for 1% Tolerance would

result in an envelope bounded by two horizontal lines at the plus and minus 1% levels across the full number of Points. The Flat algorithm thus guarantees that the transients have been settled to the specified Tolerance for some time, which tends to take longer than the Exponential algorithm.

apbAverage = Average: measurements are first discarded for the duration of the Delay interval, as with Exponential and Flat. At the conclusion of the Delay period, the number of consecutive readings specified in the Points field is accumulated and their average value computed and plotted. Tolerance and Floor values are ignored when Average is selected. The Average algorithm is particularly useful when the signal is fundamentally noisy and might never settle within a practical Tolerance.

## Appendix B

### Parameter ID# List

Using an ID# as the setting (*idnumber*) for the sweep Data 1-6, Source 1-2, and source 1 Min Level Source Selector (External Sweeps) commands is analogous to the selecting the Sweep panel Data 1 browser and choosing the desired instrument and parameter.

Example: To obtain the ID# in order to programmatically assigned a sweep parameter. Manually select the desired instrument and parameter from the desired sweep browser and note the text displayed in the selection box. Locate the text displayed in the selection box from the following list and use the associated value with the appropriate `ATS.Sweep.????.Id` command designating the desired sweep parameter to be changed.

<b>Sweep panel ID Text</b>	<b>Value</b>
Analog Generator.Burst Inteerval	5069
Analog Generator.Burst Low Level	5070
Analog Generator.Burst On	5068
Analog Generator.ChA Amplitude	5052
Analog Generator.ChB Amplitude	5053
Analog Generator.Dual Amp Ratio	5085
Analog Generator.Frequency	5051
Analog Generator.Frequency 2	5084
Analog Generator.High Freq	5087
Analog Generator.IM Freq	5054
Analog Generator.Phase	5083
Analyzer.Amplitude A	6343
Analyzer.Amplitude B	6372
Analyzer.Bandpass A	6345
Analyzer.Bandpass B	6374
Analyzer.BP/BR Filter Freq	5542
Analyzer.Crosstalk A	6016
Analyzer.Crosstalk B	6069

## Appendix B: Parameter ID# List

---

Analyzer.Frequency A	6009
Analyzer.Frequency B	6010
Analyzer.Level A	6341
Analyzer.Level B	6342
Analyzer.Phase	6365
Analyzer.Ratio A	6015
Analyzer.Ratio B	6068
Analyzer.SMPTE/DIN A	6363
Analyzer.SMPTE/DIN B	6378
Analyzer.THD+N Ampl A	6344
Analyzer.THD+N Ampl B	6373
Analyzer.THD+N Ratio A	6017
Analyzer.THD+N Ratio B	6070
Auxiliary.Reading 1 (Double)	6275
Auxiliary.Reading 2 (Double)	6276
Auxiliary.Reading 3 (Long)	6277
Auxiliary.Reading 4 (Long)	6278
Auxiliary.Setting 1 (Double)	6271
Auxiliary.Setting 2 (Double)	6272
Auxiliary.Setting 3 (Long)	6273
Auxiliary.Setting 4 (Long)	6274
DCX-127.DC Out 1	5258
DCX-127.DC Out 2	5260
DCX-127.Dig In	5953
DCX-127.Dig Out	5265
DCX-127.DMM Volts	5951
DCX-127.Gate Delay	5271
DCX-127.Port A	5268
DCX-127.Port B	5269
DCX-127.Port C	5270
DCX-127.Port D	5272
Digital Generator.Burst Interval	5129
Digital Generator.Burst Low Level	5130
Digital Generator.Burst On	5128
Digital Generator.ChA Amplitude	5106
Digital Generator.ChB Amplitude	5107
Digital Generator.Dual Amp Ratio	5132
Digital Generator.Frequency	5102
Digital Generator.Frequency 2	5115

Digital Generator.High Freq	5133
Digital Generator.IM Freq	5104
Digital Generator.Offset	5136
Digital Generator.Phase	5131
Digital Generator.Samples/Step	5135
Digital I/O.Input Jitter Ampl.	6105
Digital I/O.Input Resolution	5325
Digital I/O.Input Sample Rate	6101
Digital I/O.Input Voltage	6102
Digital I/O.Output Resolution	5326
Digital I/O.Output Sample Rate	5301
Digital I/O.Output Voltage	5304
Distortion.ChA Fund Ampl	6125
Distortion.ChA Fund Freq	6127
Distortion.ChA Harm Sum1	6129
Distortion.ChA Harm Sum2	6130
Distortion.ChB Fund Ampl	6126
Distortion.ChB Fund Freq	6128
Distortion.ChB Harm Sum1	6131
Distortion.ChB Harm Sum2	6132
Distortion.Freq	6384
Interface.Amplitude	6053
Interface.Amplitude(Prob)	5614
Interface.Frequency	5613
Interface.Jitter(Prob)	5615
Interface.Jitter(sec)	6055
Interface.Jitter(UI)	6058
Interface.Probability	6054
Interface.Time	5612
Multitone.ChA Amplitude	6310
Multitone.ChA Phase	6033
Multitone.ChB Amplitude	6313
Multitone.ChB Phase	6034
Multitone.Frequency	5621
Multitone.Resolution	5633
Multitone.Time	5620
Multitone.Trigger Delay	5634
None.None.	5049
Spectrum.ChA Amplitude	6024

## Appendix B: Parameter ID# List

---

Spectrum.ChA Phase	6065
Spectrum.ChB Amplitude	6027
Spectrum.ChB Phase	6066
Spectrum.Frequency	5515
Spectrum.Pre-Trig Time	5519
Spectrum.Start Time	5518
Spectrum.Time	5516
Switcher.ChA Input	5201
Switcher.ChA Input/Output	5206
Switcher.ChA Output	5203
Switcher.ChA+B Input	5208
Switcher.ChA+B Input/Output	5210
Switcher.ChA+B Output	5209
Switcher.ChB Input	5202
Switcher.ChB Input/Output	5207
Switcher.ChB Output	5204
Sync/Ref.In from Ref In Delay	6103
Time.Time.External Sweep Time	6253
Time.Time.Time Since Test Loaded	6251



## Appendix C

### *FFT Window Descriptions*

<b>Window</b>	<b>Description</b>
Hann	This window is a raised cosine window named after its inventor, Austrian meteorologist Julius von Hann. It provides good selectivity near the center frequency with no side lobes. Its skirts are not as steep as the Blackman-Harris window. The Hann window causes approximately a -1.5 dB maximum amplitude error due to window attenuation if the signal is at the extreme edge of the bin.
Flat-Top	This window is designed for the greatest amplitude measurement accuracy. It provides a maximum amplitude error due to window attenuation of less than 0.02 dB even if the signal is at the extreme end of the bin. However, its selectivity is poorer than either Hann or Blackman-Harris. The Flat-Top window is the appropriate window for accurate amplitude measurements (such as when measuring individual harmonics) except when signals are so closely spaced that its selectivity becomes a problem. For example, the 2.93 Hz bin width of a 16,384 sample FFT at the 48 kHz sample rate would permit accurate measurements of signals differing by nearly 90 dB in amplitude as long as they are at least 26.4 Hz (9 bins) apart
BH4	The Blackman-Harris 4-term minimum sidelobe window furnished as part of several Audio Precision FFT programs was developed by R.B. Blackman and F.J. Harris. Compared to the Hann window, it is not as selective near the nose but has steeper skirts below that point. The Blackman-Harris window has sidelobes below -92 dB (response fall-off is not monotonic). It has a reasonably flat top with a maximum amplitude error of about -0.8 dB if the signal is at the extreme edge of the bin.

Equiripple	The Equiripple window, developed at Audio Precision, is an approximation to the Dolph-Chebyshev window which provides the narrowest mainlobe width for a given maximum sidelobe depth. The mainlobe is approximately 12 bins wide; that is, the first null is about 6 bins away from the mainlobe center. The first sidelobe, which is also the highest sidelobe, is 147 dB down from the mainlobe. Maximum amplitude error across the bin is approximately 0.6 dB.
Hamming	The Hamming window has the sharpest nose selectivity of all the furnished windows. Adjacent bins average about 7 dB down and two bins away the response is about 40 dB down. Amplitude error is about -1.7 dB for a signal at the extreme edge of a bin. The Hamming window has side lobes (that is, response fall-off is not monotonic) starting only 40-50 dB below the center bin, or about 4 bins away from center. The cyan trace shows the Equiripple window, for reference.
Gaussian	The Gaussian window nose selectivity is only slightly wider than the Blackman Harris window and the near-by rejection is considerably better than Blackman Harris, reaching an average of 100 dB down in the fifth bin away from center. The side lobes are down more than 130 dB, compared to about -100 dB for Blackman Harris. Maximum amplitude error is about -0.7 dB for a signal at bin edge.
Rife-Vincent 4	Both Rife-Vincent windows have smooth, monotonically-falling responses with no sidelobes. The Rife-Vincent 4 window has about -0.6 dB maximum amplitude error, is down about -100 dB 7 bins off and about -150 dB 15 bins off. The Rife-Vincent 5 is slightly wider at the nose, with about -0.5 dB maximum amplitude error with a signal at bin edge. It has sharper skirts with attenuation reaching about 106 dB 7 bins off and about 150 dB 12 bins off.
Rife-Vincent 5	See Rife-Vincent 4 above.