APWIN BASIC

Volumes 1 through 3

User's Guide and Programming Reference

APWIN BASIC
User's Manual and
Programmer's Reference

Volume 1 Language Reference
Volume 2 Extensions A-R
Volume 3 Extensions S-Z

Version 1.52
November, 1998

Version 1.52 November, 1998

Published by:

Audio precision

Audio Precision, Inc.
PO Box 2209
Beaverton, Oregon 97075-2209
U.S. Toll Free: 1-800-231-7350
Tel: (503) 627-0832 Fax: (503) 641-8906
email: techsupport@audioprecision.com
Web: www.audioprecision.com

Printed in the United States of America
Audio Precision Part # 8211-0002

CONTENTS

Volume 1

Volume 2

Volume 3

Introduction

Welcome to the APWIN BASIC User's Manual and Programmers Reference, your guide to creating custom test programs for Audio Precision's System One and System Two platforms. APWIN Basic is a powerful and easy-to-use programming language compatible with Microsoft's Visual Basic for Applications. In this book, you'll learn how to create APWIN Basic programs (i.e. macros) that can load and run tests, automate repetitive tasks, and add custom features and functions to APWIN to suit your measurement needs.

With APWIN Macros are lists of commands that tell APWIN Basic what to do. Included with APWIN Basic are many extension commands you can use in your programs to automate control of Audio Precision's System One and System Two hardware. You do not need to develop any special commands to control APWIN and its attached hardware (System One or System Two); all of these commands are available when you begin using APWIN Basic.

One of the most exciting features in APWIN Basic is its support of OLE automation. OLE stands for Object Linking and Embedding and is a standard used in Microsoft Windows to allow OLE compliant applications to share information. Using the OLE automation features in APWIN Basic it is possible, for example, to take the results from a measurement made on System One or System Two hardware, move the data into any Excel spreadsheet where it can be further manipulated, then take these results into Microsoft Word where they can be inserted into a standard report form. All of this can be automated and run entirely within APWIN Basic. The results of your Word document can even be printed from inside APWIN Basic.

All of this power and functionality might lead you to think APWIN Basic is a difficult and complex programming language. In fact, APWIN Basic is one of the easiest development environments to use. Even if you have never programmed before, you will be surprised how quickly you will begin developing interesting and powerful programs.

How This Book is Organized

The chapters of the *APWIN Basic User's Manual and Programmers Reference* are arranged in three sections and three volumes.

Volume One

Section One provides an introduction to programming in APWIN Basic. It is intended as a tutorial to help beginning users understand what APWIN Basic is and how to use it to develop programs. Depending on your experience programming with Visual Basic, you may want to read some or all of these introductory chapters. Section One is covered in chapters 1 - 6.

Section Two is organized as a Language Reference and lists the generic commands available in APWIN Basic. These are the same commands you will find available in any Visual Basic compatible application. The Language Reference is covered in chapter 7.

Volume Two

This volume starts section three and contains detailed descriptions of the extension commands A through R in APWIN Basic.

Volume Three

This volume finishes section three and contains detailed descriptions of the extension commands S through Z in APWIN Basic.

Extension commands are used to control the operation of APWIN and System One or System Two hardware. These commands, along with several examples showing how to use them, are discussed in chapter 8 through 42.

Chapter Overviews

- Chapter 1 provides an overview of APWIN to help the first time user get started quickly. The first time user should review this chapter before continuing.

- Chapter 2 provides an introduction to the fundamentals of APWIN Basic. Several of the key concepts in Visual Basic are introduced, including objects, methods and properties, and the use of macros.

- Chapter 3 moves beyond the concepts of Visual Basic and jumps into the basics of writing a program. Working from a simple example, each of the key elements of a program is introduced and discussed. Some of the key topics discussed in this chapter include the structure of a program, syntax, and an introduction to commonly used commands.

• Chapter 4 describes how to test and debug a program. APWIN Basic provides a number of tools to assist in verifying correct operation of a program. Additional topics include tips for simplifying the debugging process, common programming mistakes to avoid, and error handling.

• Chapter 5 provides an introduction to the APWIN Basic Dialog Editor. The Dialog Editor provides an easy way of creating a user interface consisting of menus, and other dialogs that an operator can interact with to control your program.

• Chapter 6 provides an introduction and an example of how to interact with other applications using OLE to produce custom reports.

Manual Conventions

This manual uses the following typographic conventions.

Example	Description
event, var, arg	For the syntax part of each command, italicized words indicate placeholders where the user must enter additional information.
FILENAME.TXT	Words in all CAPITOL letters indicate file names.
Sub Main AP.Gen.Amp = 1.0 End Sub	This font is used in all example macros and code modules.
[*expressionlist*]	In syntax, items inside square brackets are optional.
{*While* | *Until*}	In syntax, braces and a vertical bar indicate a choice between two or more items.

Command

For the syntax part of each command, the bold characters identify the part of the command that must be entered.

```
AP.Prompt. _
   Text “This _
   is just an _
   example.”
```

The line continue character (_) is used to indicate that the code from one line to the next should be typed on one line.

❶ This symbol denotes a command or example procedure that is applicable to System One.

❷ This symbol denotes a command or example procedure that is applicable to System Two.

A Few Words About Terminology

Audio Precision has used the term Procedure since the first product to identify a facility that will automatically run a sequence of tests. This term has in fact been used for many years in the industry to generally describe the process of performing one or more tests and/or measurements. A Test Procedure has described what to measure, how to measure it, what equipment to use and other details that a technician would need to know to carry out the task in a consistent fashion that meets the objectives of the test.

Programmers have also used the word Procedure for several years to identify specific programs or parts of programs. In particular, Visual Basic and Applications Basic uses the term Procedure to mean a specific part of a program. Unfortunately, this use of the word is at odds with the testing industry use of the term Procedure as described above. The programming world uses the term Macro to describe what test engineers call a Procedure.

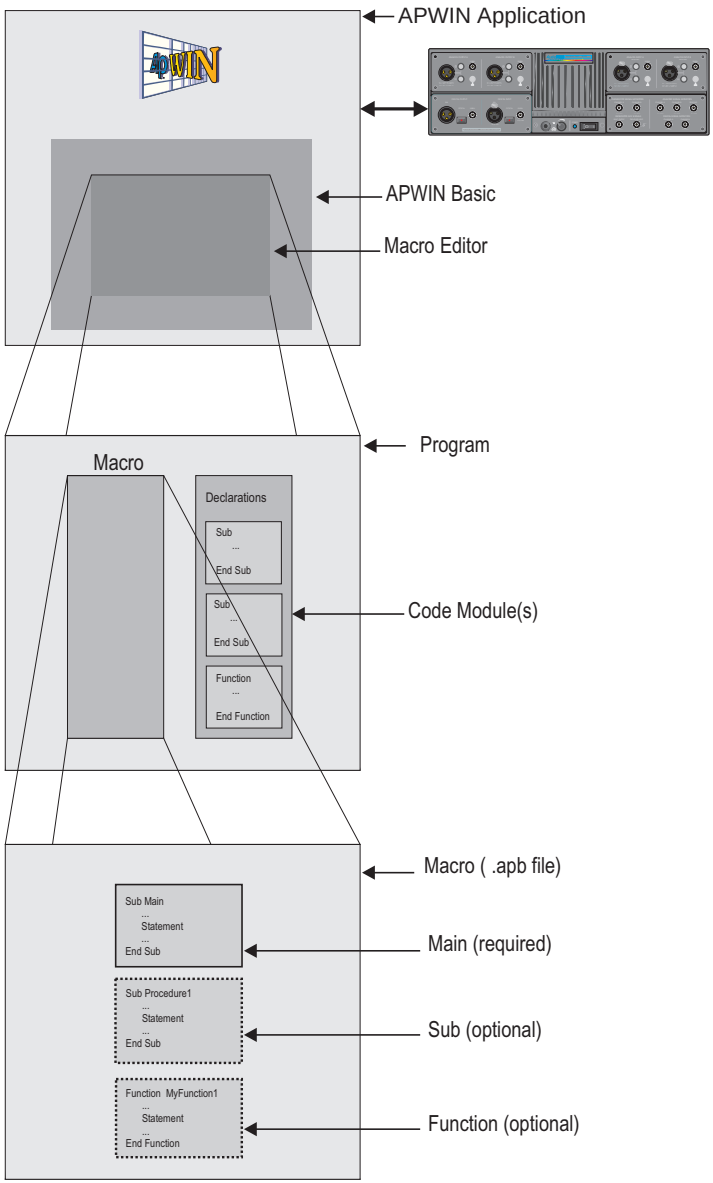


Figure 1-1

In reference to APWIN BASIC and for purposes of this manual the term Procedure refers to Sub and Function Procedure parts of a macro or code module as shown in figure 1-1.

Where to Find Sample Files and Examples

Included with your APWIN software are sample programs for both System One and System Two hardware. If you choose to include the samples as part of your APWIN installation, they can be found under the APWIN subdirectory. Examples for System Two are found under the S2 subdirectory and examples for System One are in the S1 subdirectory. The directory structure for the sample files is shown in figure 1-2.

These examples are excellent learning tools and are representative of the type of programs your are likely to develop. You can load these macros into the APWIN Basic editor where you can edit any part of them or even copy them into your own program.

Samples contained in the DEMO directories are designed to be run without System One or System Two hardware attached.

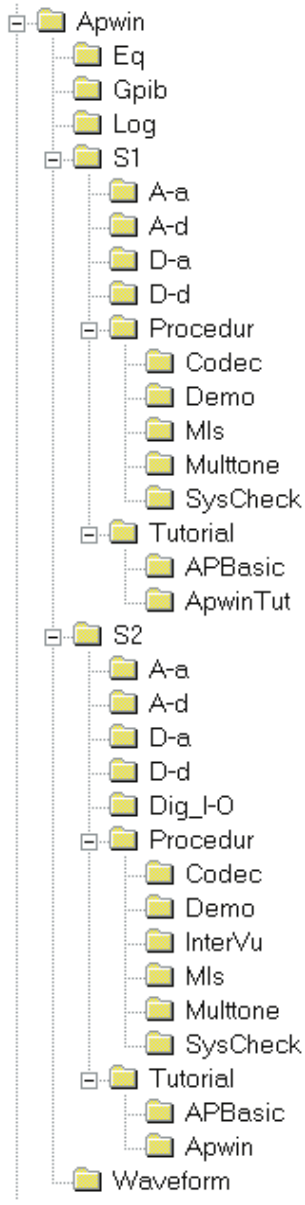


Figure 1-2 Default APWIN directory structure.

Using Online Help

APWIN provides extensive online help to assist you in developing APWIN Basic programs. Help is accessible in the following ways:

• Choose the Help menu in APWIN. If you have already opened the Macro Editor panel, you can select between APWIN Basic Language, APWIN Extensions, or the APWIN Basic Editor Help. If you have not opened the Macro Editor panel, these help options will not yet be available to you.

• Highlight a command or keyword in the APWIN Basic Editor and press F1 for context-sensitive help.

• Select the Browse Object icon  on the APWIN Basic Editor toolbar, and then select the method or property you need information about. The Object Browser provides information about all of the classes and objects available in APWIN.

• Highlight a specific APWIN Basic extension command and press the Browse Object icon on the APWIN Basic Editor toolbar for information about the methods and properties of the command.

Getting Started In APWIN Basic

APWIN Basic is automatically installed on your computer when you install Audio Precision's APWIN software. There are no extra installation steps necessary to use APWIN Basic. See the Audio Precision *APWIN Getting Started Manual* for instruction on installing APWIN.

To begin using APWIN Basic, open the Macro Editor panel inside of APWIN. You can open this panel by selecting the *Macro Editor* option under the *Panels* pull-down menu or by pressing the Macro Editor icon  on the icon toolbar.

Figure 1-3 contains a picture of the macro editor panel after it has just been opened.

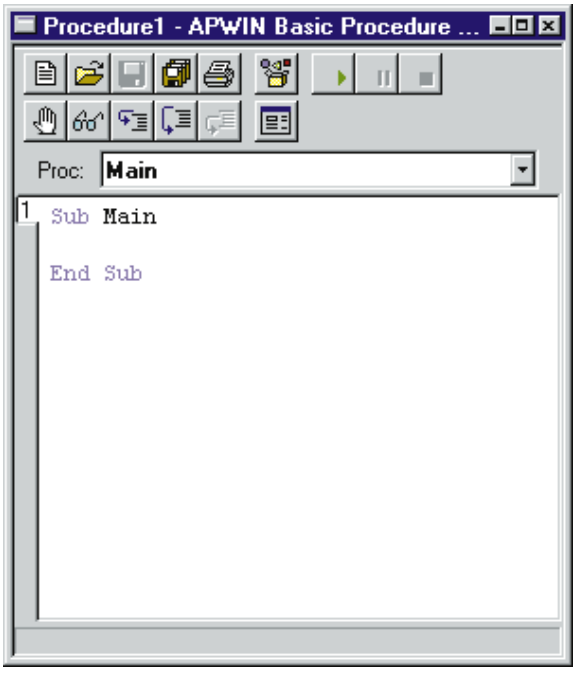


Figure 1-3

APWIN Basic Editor Overview

The APWIN Basic Editor offers a number of menu options and buttons to make it easier to use. Each of the buttons available is explained in detail in the on-line help section of the APWIN Basic Editor. You can also get information about a specific button by leaving the mouse pointer over a button for a few seconds. A short information bar "tool tip" will pop up indicating the purpose of the button.

The menu options are made available by clicking the right mouse button once in the main editor window shown below in figure 1-4.

Click the mouse on any of the available menu options to select the option you want. To open an APWIN Basic macro, select either the open icon button,  or the *File* menu option.

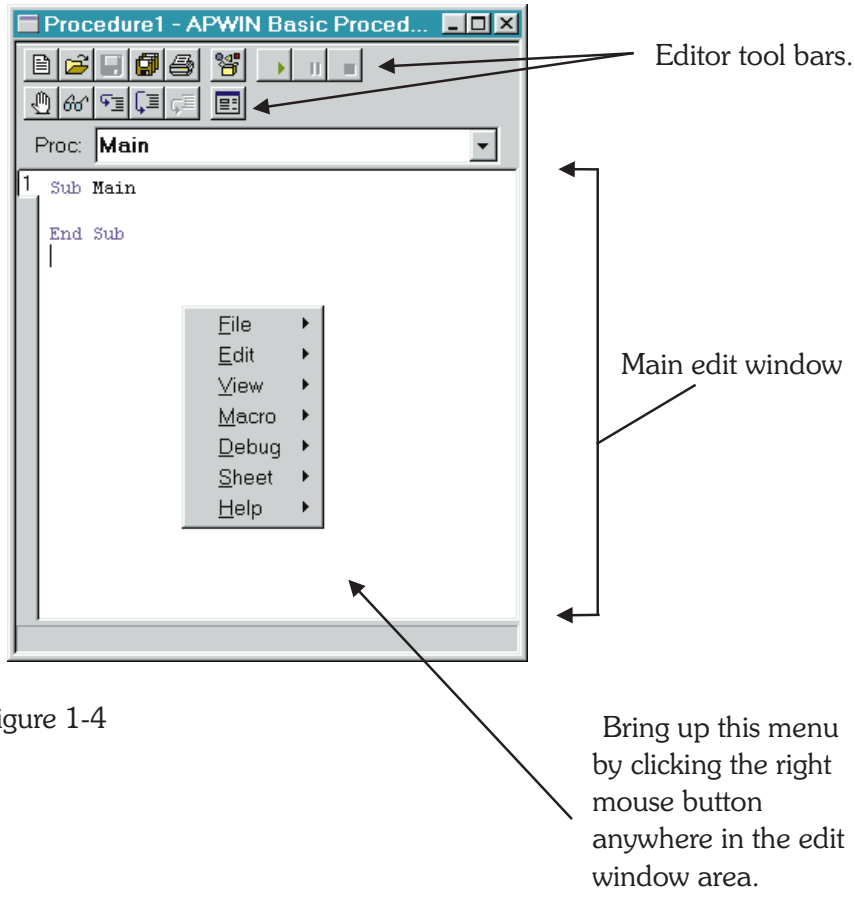


Figure 1-4

You can use the APWIN Basic Editor to open several macros at one time. Each time a macro is opened, a new sheet is created and the macro is placed on the sheet. You can select between sheets by pressing the number on the sheet toolbar corresponding to the macro you want (See figure 1-6). This allows you to quickly switch between macros when you want to cut and paste code. You close a sheet by double-clicking on the sheet number or selecting the sheet menu option and then selecting close.

Once you have loaded or entered a macro, you can run your macro by selecting the Macro Run button from the icon toolbar as shown in figure 1-5.

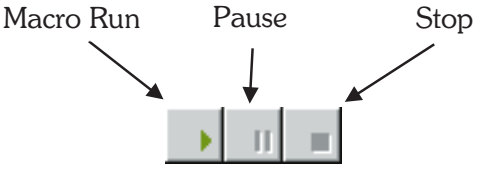


Figure 1-5

When you run a macro, APWIN Basic will execute the commands that make up the macro. If you have several sheets open at one time, APWIN Basic will only run the macro that is currently shown when the Macro Run button is pressed. To execute a different macro, you must first select the sheet number using the sheet toolbar shown below in figure 1-6.

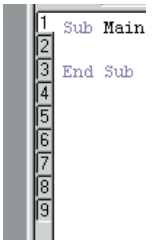


Figure 1-6

The APWIN Basic Editor can also be used to diagnose and fix errors in your program. To use the debug features you can select among the debug buttons along the icon toolbar  or choose debug from the menu options. The topic of testing and debugging code is discussed in more detail in chapter 3.

Editing Code with the APWIN Basic Editor

To edit or enter new code with the APWIN Basic Editor, use the mouse to position the cursor to where you want your code to begin and start typing. You will find the APWIN Basic Editor operates much like other word processing editors available in Microsoft Windows. You can cut, copy and paste code using the CTRL-X, CTRL-C and CTRL-V hot keys that are standard in Windows, or you can select cut and paste from the edit menu option. It is also possible to copy text from a different Windows application and paste it onto a sheet. For example, you can copy sample code fragments from the APWIN Basic Help screen and paste these into your program. Figure 1-7 shows several lines of code that have been highlighted. The highlighted code

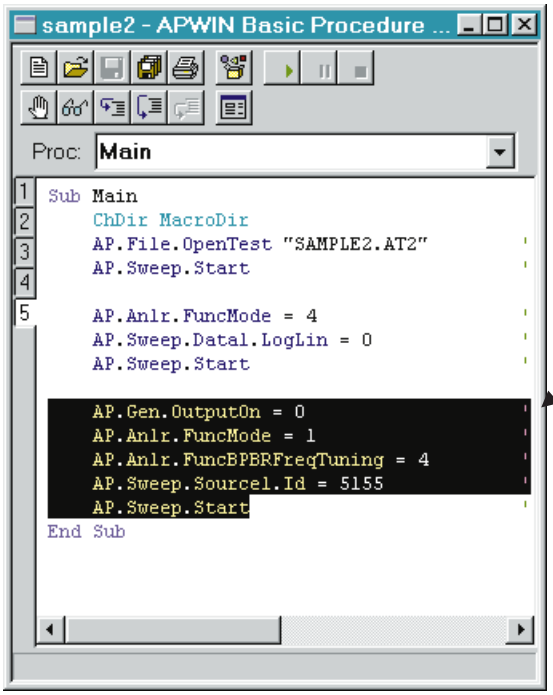


Figure 1-7

can be copied or cut from the current location and inserted (pasted) at another location.

To learn more about editing code or how to use the APWIN Basic Editor in general, refer to the online help under APWIN Basic Editor.

Where to Find More Information About Visual Basic

There are several good references available to help you learn Visual Basic. We recommend you consider the following:

- The *Running Visual Basic for Windows* by Ross Nelson
- The *Microsoft Visual Basic Programmer's Guide*
- The *Microsoft Excel Visual Basic User's Guide*
- The *Microsoft Word Developer's Kit 6.0* This manual provides command reference and other information needed to communicate with Microsoft Word in order to produce reports or other documentation via OLE automation

Information for Experienced Visual Basic Programmers

As an experienced Visual Basic programmer, you may need to understand how APWIN Basic differs from Visual Basic. To answer this, we need to look at the different types of Visual Basic. Currently, Visual Basic exists in three editions: a Professional edition, a Standard edition, and an Applications Edition. The Professional and Standard editions are the original stand-alone programming environments used to create complete Microsoft Windows based applications. The applications edition of Visual Basic, known as Visual Basic for Applications or VBA, is a subset of Visual Basic. VBA is designed to be embedded within an application and does not offer the forms package available in the Professional and Standard editions. APWIN Basic is compatible with VBA. It does not include the forms package and can only be run from within APWIN. For information on specific APWIN Basic commands that may differ from standard Visual Basic, consult the online help.

User Notes

User Notes

Fundamentals of APWIN Basic

This chapter is the first of several chapters that introduce APWIN Basic. We begin our discussion with procedures, one of the most basic elements in an APWIN Basic program. Procedures are used to group commands together that when combined perform a specific task. Collections of procedures are often organized to form a complete macro. We will look at how procedures are structured and how they are used in programs.

In the second half of this chapter, we shift from procedures and study how data is represented in APWIN Basic. *Objects* are introduced as a way to organize collections of code and data that are related. *Properties* are characteristics of objects that can be used to change the attributes of an object. *Methods* are another characteristic of objects that can perform a function. Although procedures and objects may at first seem to be related in how they group together common commands and data, they are distinctly different parts of a program. In this chapter we will examine procedures and objects more closely.

A complete discussion of the different parts of a program is postponed until chapter 3. If you discover while reading this chapter that you need an example of a program to work from, you can flip to the beginning of chapter 3 where a complete program example is given.

What is an APWIN Basic Program?

A *Program* is a collection of one or more APWIN Basic *Macros*. Each *Macro* can contain zero or more *Procedures*. Each *Procedure* contains commands that do something useful.

For example, a program might be written to load and run a number of tests in APWIN. Another program might be written to combine the results of several tests and extract common trends in the data. Yet another program might offer a dialog box from which a user can select between different programs to run. There is no requirement on what a

program must do other than it must consist of legal APWIN Basic commands that can be executed.

A program can be as big or as small as you choose. Since programmers often want to combine several different operations into one program, programs tend to become large and complex fairly quickly. *Procedures* are used to help organize programs into sections of similar code.

Using Procedures

Procedures are collections of APWIN Basic commands that are executed as a unit. When APWIN executes a procedure, it starts with the first command in the procedure and proceeds from top to bottom, one line at a time. A well written procedure should accomplish a single task. For example, a procedure might load and run a test, alter how APWIN is configured, or collect information from a user. Complicated tasks should be brokendown into several sub procedures. A complete program may use any number of procedures.

There are three main benefits of programming with procedures.

- Procedures allow you to break your application into separate, logical elements, each of which you can understand and debug more easily.

- Procedures can simplify and condense code by combining repeated or common tasks into just one piece of code.

- Procedures used in one program can be copied and used as building blocks for another program. Once you have a procedure that works well, you will want to use this procedure in other programs rather than spending the time to re-write code.

APWIN Basic uses two main types of procedures: *sub* procedures and *function* procedures. A sub procedure performs a specific task but does not return a result. A function procedure is similar to a sub procedure except that it can return a result. Each of these types of procedures is discussed in more detail below.

Elements of a Procedure

Before exploring the differences between sub and function procedures, it's instructive to look at the elements common to all procedures. A clear understanding of a procedure's structure will help you avoid common mistakes that often frustrate beginning programmers. It will also help you to read and understand other examples of APWIN Basic code.

All procedures have the following parts:

- Begin and End statements at the top and bottom of the procedure, respectively
- A label that uniquely identifies the procedure
- Arguments that follow the procedure label
- APWIN Basic code

The beginning and end statements for a *sub* procedure follow the general form:

```
Sub ProcedureName (arguments)
   …
End Sub
```

The first line of a sub procedure always begins with the Sub statement, the name of the procedure, and a set of parentheses in which arguments are placed. If the procedure doesn't require any arguments, the parenthesis are not required. The label of a procedure is a unique name you choose that allows you to refer to the procedure. Typically, you should choose procedure labels that describe what the procedure does. For example, a procedure that prompts the user for their initials might use the following first line:

```
Sub PromptForInitials ()
```

A procedure label can be almost any combination of characters and numbers except that it must start with a character and not contain any spaces.

The arguments that follow a procedure label allow the programmer to pass specific information to the procedure. During a typical program, a procedure may be executed from several different points in the code, but the data used by the procedure may need to change. Arguments provide a means to vary the information used in a procedure. The topic of arguments and how and when to use them in procedures is not difficult but has some subtleties and variations that are beyond the scope of this tutorial. Refer to any of the Visual Basic programming manuals mentioned in the introductory chapter for more information on using argument in procedures.

The bulk of a procedure consists of the code. These are commands that tell Basic what to do. There are a large number of commands available in APWIN Basic and almost all of them may be used in procedures. Any command you want to use in a procedure must be placed within the `Sub` and `End Sub` statements.

Technically, the number of commands you can place in a procedure is quite large; practically, however, you will want to limit the number of commands in any one procedure. Your goal when writing a procedure should be to use only the commands you need to accomplish a specific task. If your program needs to do several different tasks, then you should write several different procedures, one for each task. It is much easier to understand and debug small blocks of code than to try and sift your way through an unnecessarily large and complex procedure.

The second type of procedures used in APWIN Basic are *function* procedures. They are similar to *sub* procedures and follow the general form:

```
Function FunctionName(arguments)
   …
End Function
```

Function procedures are written in the same way as sub procedures but with one important difference. The commands inside a Function should assign a return value to the name you gave the function. When the function is finished executing, APWIN Basic will return the value assigned to the function name to the line of code that called the function procedure.

For example, you could write a function that calculates the value of a number in decibels (dB).

```
Function TodB (num)
    TodB = 20*Log10(num)
End Function
```

You call a Function procedure the same way you call any of the built-in functions in APWIN Basic.

```
result = TodB (data)
```

Here is the previous example together with sample code that calls the function procedure. In this example, two channels of data are converted, one element at a time, to a dB format.

```
Sub convertData(numPoints)
  For n = 0 To numPoints
    dataCh1(n) = TodB(dataCh1(n))
    dataCh2(n) = TodB(dataCh2(n))
  Next n
End Sub

Function TodB (num)
    TodB = 20*Log10(num)
End Function
```

The techniques for calling all types of procedures are discussed in the section *Calling Procedures*, later in this chapter.

Sub and Function procedures are the building blocks of any APWIN application. They can be combined and used in any way you choose to make your application useful. The next section looks more closely at some of the different ways to use procedures.

How to Use Procedures

In order to develop an APWIN Basic program, you must first understand how to use procedures. In this section we look at some of the different uses of procedures and how they can be combined to form a menu.

One key use of procedures is to define where program execution begins. A typical APWIN Basic program may have several different sub and function procedures. In order to begin running the program, APWIN Basic must know which of these to start from.

In APWIN Basic, program execution starts with the first line of code in the *Main* sub procedure. The *Main* sub procedure is just like any other procedure. You can use any commands you want in any order you choose. What's special about the Main sub procedure is that execution will always start with the first line of code. Here is an example of a Main sub procedure.

```
Sub Main
   Call runTest()
   Call processResults()
   Call printResults()
End Sub
```

In this example, the only code in the Main sub procedure are calls to other sub procedures. In this way, the Main sub procedure is used to organize how program execution flows through the code.

All APWIN Basic programs you write will need to have a Main sub procedure. If you try to run your program without a Main sub procedure, or with two sub procedures using the Main label, you will get an error.

Unless your program is very simple, you're likely to want to use several procedures in addition to the Main sub procedure. As shown below, you access additional sub and function procedures by *calling* them from within another procedure.

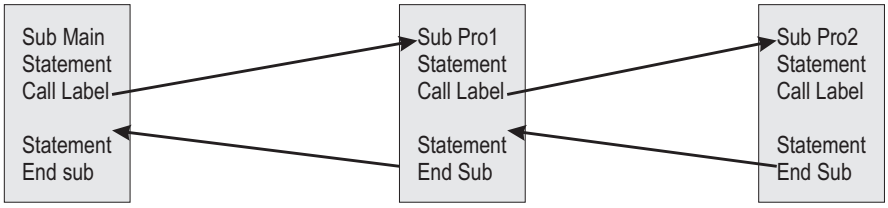


Figure 2-1

Calling Procedures

The techniques for calling procedures vary, depending on the type of procedure, where it's located, and how it's used.

A *sub* procedure is called by a stand-alone statement on it own line of code. Unlike a function procedure, a sub procedure does not return a value, but can modify the values of any variables passed to it.

There are two ways to call sub procedures.

```
Call MyProcedure (argument1, argument2)
```

-OR-

```
MyProcedure argument1, argument2
```

Note that when the Call syntax is used, the arguments passed to the procedure must be enclosed in parentheses. When the Call syntax is not used, the parentheses can be omitted.

A call to a function procedure is made in the same way you call any intrinsic Visual Basic function, like `Log10`, that is, by using its name in an expression.

```
‘ The following statement calls the TodB function
result = TodB (data)
```

It is also possible to call a function procedure just like you would a sub procedure.

```
Call TodB (data)
```

-OR-

```
TodB data
```

When functions are called this way, APWIN Basic throws away the return value.

Shown in figure 2-2 is an example of an APWIN Basic program that calls two different sub procedures. Note how program execution returns from each called sub procedures.

Calling Sub and Function Procedures From Other Code Modules

A procedure can also be called from another macro or code module. It is possible to call procedures in other macros from anywhere in your program.

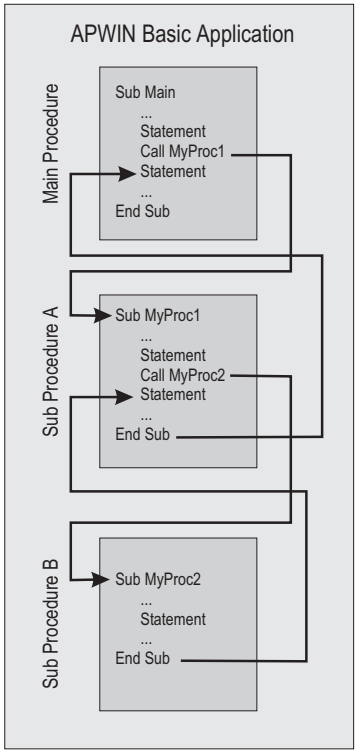


Figure 2-2

To call a sub or function procedure in another macro, also known as another *code module*, you must include a reference to the code module in your macro. You make the reference to the code module with the `'#uses` statement. The `'#uses` statement has the following syntax.

```
'#uses "MODULENAME.APB"
```

There are several important steps you must follow to use the `'#uses` statement correctly.

- Make sure to include the "`'`" character in front of the "#" character.
- Add the `'#uses` statement on the first line of your program
- Include the path to the code module you want to include within the quotes if the code module exists in another directory.

Note that the "`'`" character is normally used to add comments to your code. It is needed here since the `'#uses` statement is not a normal APWIN Basic command and is not compatible with Visual Basic which uses another form of include.

When you add the `'#uses` statement to your macro, all of the sub and function procedures of the code module are available to your macro. You call these included procedures just as you would a normal procedure.

The following line of code will include all of the sub and function procedures of `MYDEMO.APB` in your program.

```
'#uses "C:\APWIN\DEVELOPMENT\MYDEMO.APB"
```

One reason for including sub and function procedures from other code modules is that you can create a library of commonly used procedures. Once you have a library, any program that wants to use a library procedure just needs to include the appropriate `'#uses` statement.

To learn more about including procedures from code modules in your program, refer to the online help.

Introduction to Objects, Methods, and Properties

In this section we shift from an introduction to procedures and present some of the more conceptual ideas behind Visual Basic. Much of this conceptual framework centers around how data is represented. For

those of you who are new to object oriented programming, or are new to programming in general, these ideas may seem strange and even confusing. Fortunately, it is not necessary for you to master this section to begin developing APWIN Basic programs. Instead, the concepts introduced here are intended to expose you to some of the vocabulary and ideas which more experienced programmers use when working with Visual Basic.

What Are Objects?

An *object* is a combination of code and data that can be treated as a unit. An object may be a part of your program or even the entire program. An object may even represent something physical, like the analog generator of a System Two. Almost anything you want to represent in Visual Basic, either real or imaginary, can be expressed as an object.

Some examples of objects available to you in APWIN Basic are described in the table below.

Example	Description
Dialog Box	A dialog box that reports information to the user or prompts the user for data is an object.
Chart	A chart in Microsoft Excel is an Object
Database	Databases are objects that can contain other objects, like fields and indexes.
System Two Hardware	Audio Precision's System Two is represented in APWIN Basic as a library of objects that are contained in the AP *class*.

Objects are used in APWIN Basic to make your work as a programmer easier. Since objects can represent complex data structures and code, they can simplify your program by allowing you to use them rather than requiring you to write your own code. For example, you could write your own code to create a chart similar to one you might find in Microsoft Excel, but you don't have to. Instead, you can use Excel to

create your chart and then you can manipulate it with the properties of the Chart object.

Usually, when you develop programs in APWIN Basic, you will only need the objects that are already provided as standard pieces of Visual Basic and APWIN Basic. However, it is also possible to create your own objects to simplify your code. For more information on creating your own objects refer to any of the suggested texts mentioned in the section, *Where to Find More Information About Visual Basic.*

There are three things you can do with objects in APWIN Basic that make them useful.

- You can set the *value* of an object's *property*
- You can return the *value* of an object's *property*
- You can use a *method* of the object to perform a task

In the last few sections of this chapter we look more closely at how to use properties and methods to change and control objects.

Working With Objects

Objects in APWIN Basic support *properties, methods,* and *events.* The settings and attributes of an object are called its *properties*, and the procedures that operate on an object are called its *methods*. An event is an action, like pressing a key or clicking the mouse, that is recognized by an object. You can write code to control how an object responds to an *event*.

Properties of an Object

Properties are special attributes of an object. You use properties to control the appearance of an object, its behavior, or both. A property

has a value associated with it that can be read to learn about the condition of an object or set to change the object. For example, an object may have an *enabled* property you set to **True** to activate the object. To turn Channel A of the analog generator *on* you would use the APBASIC extension command:

```
AP.Gen.ChAOn = True
```

To turn the generator off, you set the property to **False**. Sometimes, you may need to know the value of a property without wanting to change the property. To determine the value of property without changing it you assign the value of the property to a variable:

```
variable = AP.Gen.ChAOn
```

You can now test the variable without altering the property. An alternate way to check a property without changing it is to test the property in more complex expression.

```
If AP.Gen.ChAOn = True Then
   AP.Gen.ChBOn = True
Else
   AP.Gen.ChBOn = False
End If
```

Some objects may also require a parameter be specified to determine the value of a specific property. For example, to determine the amplitude of channel A on the analog generator of System Two you would use the statement:

```
variable = AP.Gen.ChAAmpl (“V”)
```

The (“V”) parameter tells APWIN Basic that you want the answer to be specified in volts.

Objects often have several properties, some of which may be common to more than one object, while other properties are unique to a single object. A specific set of properties and methods are what makes one object different from another object.

Using the Methods of an Object

Methods are another characteristic of objects. When you use a method associated with object you make the object perform a specific task. To call a method, you use the object name and the method name, separated by a period. For example, using APWIN Basic code you can open a previously saved System Two test using the *OpenTest* method associated to the *File* object in the *AP* class.

```
AP.File.OpenTest “analog THD measurement.az2"
```

An object may have a number of different methods associated with it. An example of using a second method associated with the *File* object is:

```
AP.File.OpenWfm “ISO 31 tone generator waveform.aas”
```

Like properties, methods are part of what defines an object. They are useful because they allow you to perform specific tasks without having to write the code yourself.

Using the Object Browser to Learn More About Objects

APWIN is filled with objects you can use in your APWIN Basic code. To help you search through all the available objects to see what might be useful to you, APWIN Basic provides a special dialog box called the Object Browser. You can open the Object Browser by pressing the

 icon on the Procedure Editor panel. Figure 2-3 shows what the Object Browser looks like:

The Object Browser is a source of useful information about the objects and the code in your application. You can use the Object Browser to learn more about:

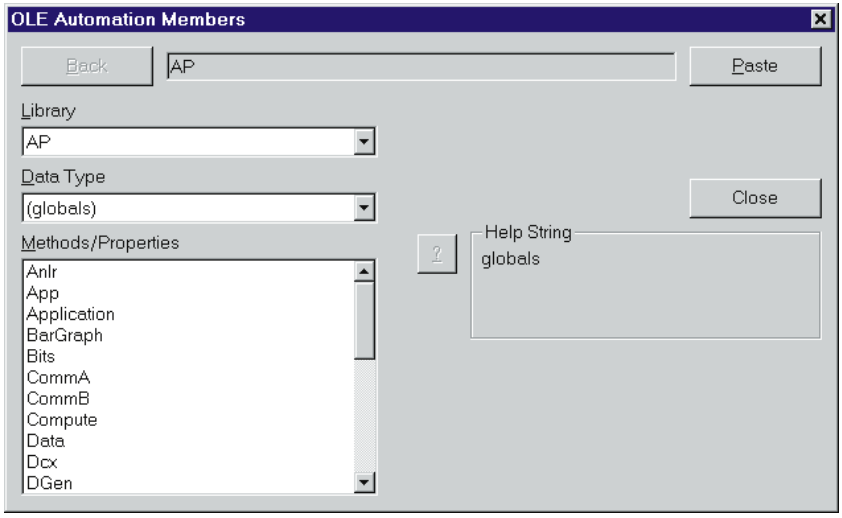


Figure 2-3

- The OLE object libraries available to you
- The names of all the objects in a given object library
- The name of all the methods and properties for any object
- The parameters for a particular method or property

In addition to the information displayed by the Object Browser, it can also be used to insert an object and its appropriate method or property, directly into your code. When you double-click on a method or property in the Object Browser, it will be inserted into your code where the cursor is placed.

All of the methods and properties available in the Object Browser are discussed in greater detail in the Extensions Reference section of this manual.

User Notes

User Notes

Writing An APWIN Basic Program

Chapter 1 introduced the fundamentals of APWIN Basic. The theory of procedures, objects, methods and properties were discussed in Chapter 2 along with simple examples to familiarize you with the key concepts of Visual Basic programming. Here, in chapter 3, these concepts are applied to create an APWIN Basic program.

A complete program is written with a specific structure and uses keywords and commands to accomplish tasks. Using a simple program as an example, we will examine what pieces are necessary in an APWIN Basic program. Some of the key topics discussed include:

- converting DOS S1.EXE procedures to APWIN Basic Macros
- using Learn Mode to enter commands directly into your code
- program structure
- adding comments to your code
- keywords and commands
- creating and declaring variables and constants
- using conditional statements to control program flow

Converting S1.EXE Procedures to APWIN Basic

For those familiar with the DOS S1 procedure language, APWIN Basic is a significantly different approach to creating automated test processes. Since it uses a subset of Visual Basic, it is much more flexible, and can be easily integrated with other software. But greatly increased capability and flexibility come at the cost of more complexity.

The basic paradigm of APWIN procedures is the same as DOS S1: sequences of stored test setups are loaded and run, and the results are displayed in graphs, tables, or reports. Most of the common commands in DOS S1 (LOAD, SAVE, RUN, COMPUTE) have direct analogues in AP Basic. DOS S1 tests, limit files, sweep files, etc. can all be converted into equivalent APWIN files. In general, APWIN Basic is a superset of the DOS S1 procedure language.

DOS S1 to APWIN Procedure Converter

By selecting Import S1.EXE Procedure from the File menu the process of translating a DOS S1.EXE procedure file (*.PRO) into an APWIN Macro (*.APB) is initiated. In addition, it imports all the DOS S1.EXE tests, limit files, sweep files, EQ files (*.TST, *.LIM, *.SWP, *.EQ), etc. that are required by the procedure, and saves them in their equivalent APWIN form. Thus, it is an efficient method of converting a DOS S1.EXE procedure and associated files into an APWIN macro.

• Directory structure - In its simplest form, the converter expects the DOS S1.EXE procedure and all its tests, etc. to be together in one folder (directory). You specify this folder in the first File Open dialog when you choose the procedure file (*.PRO) to convert. The second dialog (File Save As) specifies the folder where the new procedure file (*.APB) and its tests, etc. will be saved. Usually, this is also a new folder, to help keep old and new files segregated. If your DOS S1.EXE procedure includes a DOS Change Directory (Chdir) command, the converter will recognize this and ask you whether you want to specify a different folder in which to save the new files as well.

• Multiple procedures - The converter only translates one procedure at a time. If you call sub-procedures or you string several procedures together, you must convert each of these separately before you can run the new procedure.

• Conversion problems - Most converted procedures will not run immediately after conversion. This is because not all DOS S1.EXE commands translate directly into APWIN Basic commands. Some require more information, some are implemented by a different method, and some have no equivalent in APWIN Basic. If the converter cannot completely translate a command, it goes as far as it can, and alerts you to the problem by adding a comment to the line of code. (Comments in APWIN Basic are preceded by an apostrophe, and are highlighted in green by the editor. They are for reference only, and are ignored when the program is executed.) Any command which cannot be translated at all is added as a comment line to the new program exactly as it appeared in the old program. In addition, to aid in debugging, you are also given the option of adding all the

original commands as comments to the new program. After the conversion is completed, APWIN will warn you if any problems were encountered. If so, you must edit the new procedure to complete or correct any of these translation problems.

• S1 Panel moves - Because DOS S1 panel field selection uses both keyboard shortcuts and cursor movement, it is not always possible for the converter to determine which field was selected. This makes automatic translation impractical unless only the keyboard shortcuts were used. In general, for Panel commands, you will need to run the DOS S1 procedure and determine what change was made to which field. You can then use Learn Mode to generate the appropriate command for the same change in APWIN.

• Converting Overlays - Overlay files (.OVL) are used in DOS S1 procedures to perform subsequent tests using the setup of an initial test (.TST) file. Fields in the overlay panel are selectively "punched out", so that these fields are not changed when the overlay is loaded, thus preserving the value set in the initial test.

Overlays can be divided into two classes: those with a few fields punched out, and those with most fields punched out.

An overlay with only one or a few fields punched out is intended to preserve a value(s) of the underlying test. For instance, you might set a dBr reference level in Test1 and then load Overlay2 with this field punched out, so that it will use the same reference level as Test1. In APWIN, this is accomplished instead by using two tests (Test1, Test2), saving the value of the reference level to a variable before you load Test2, and then setting the reference level of Test2 to this variable. You can identify the appropriate APWIN Basic command for reading and setting the reference level by using Learn Mode.

An overlay with most fields punched out is used to change only a few items from the previous test. In APWIN, you simply make changes to the desired fields directly with APWIN Basic commands. Again, you can use Learn Mode to write these commands for you.

The converter does not import DOS Overlay files! Instead, you must do one of the following before you run the procedure converter:

For the case of an overlay where only a few fields are punched out, you should make note of these particular fields, and then save the

overlay as a test. Edit your DOS S1 procedure to read 'LOAD TEST' rather than 'LOAD OVERLAY'. The converter will then translate this properly, and will convert the DOS S1 test to an APWIN test. After conversion, you will need to edit the APWIN procedure to reset the values of the punched out fields to what they were in the previous test.

For the case of an overlay with most fields punched out, you should make note of the remaining fields and the values they are set to. Edit your DOS S1 procedure to delete the 'LOAD OVERLAY' commands since you will be adding these manually. After conversion, you will need to edit the APWIN procedure to add APWIN Basic commands to set the values of the fields you noted from the overlay.

• BarGraphs - The procedure converter will generate APWIN Basic commands to display bargraphs for the Data1, Data2, and Source1 fields on the sweep panel. This is consistent with the operation of RUN BARGRAPH (F2) command in DOS S1. However, it is preferable to save the bargraph settings with the test setup in APWIN, since you can size and position them as you choose, and set their properties separately from the sweep panel. You should do this after conversion by running the new APWIN procedure, and pausing it when a bargraph is displayed. You can then arrange the bargraphs as you see fit, and save them with the current test.

• DOS and XDOS - These commands are translated using the APWIN Basic 'Shell' command or an equivalent APWIN Basic command (ie: Dir, Chdir, etc.). It is important to note however, that there is a fundamental difference in the operation of the Shell command from the DOS command. Because Windows is a multi-tasking operating system, the Shell program runs as a separate task in parallel with APWIN. In other words, the APWIN Basic program does not wait until the Shell command is complete to continue execution. If your program requires this type of operation, please refer to the following example.

Example:

```
Private Declare Function WaitForSingleObject Lib _
    "kernel32" (ByVal hHandle As Long, ByVal _
   dwMilliseconds As Long) As Long
Private Declare Function OpenProcess Lib "kernel32" _
   (ByVal dw As Long, ByVal bInherit As Long, ByVal _
   dwProcessId As Long) As Long

Sub Main
   TaskId = Shell ("????????.???",2)
   Process = OpenProcess(&h1F0FFF,0,TaskId)
   WaitForSingleObject(Process,-1)
End Sub
```

• UTIL PROMPT with /C10 - This combination is used in DOS S1 to accept user input. The prompt is followed with IFn[commands to determine which choice was made. The procedure converter translates this to a prompt and an Input Box, and the appropriate If statements. While this works and is faithful to the DOS S1 paradigm, there are more elegant ways to accept user input using the 'Dialog' function in APWIN Basic.

As a final note, you should keep in mind that the procedure converter is intended to produce a working translation of your existing DOS S1 procedure. Toward this end it should be successful for the majority of DOS S1 procedures. However, this this working translation should only be considered a starting point for APWIN conversion, because it is only intended to duplicate DOS S1 functionality. It does not take advantage of the multitude of new alternatives that APWIN offers for test techniques, data display, user input, and program flow control.

Using Learn Mode

New procedures, or additions to existing procedures, may be generated by two different techniques. One method, suitable for those with some experience with programming techniques and knowledge of the specific syntax and commands of APWIN Basic or other forms of Visual Basic, is by typing and modifying text in the Macro Editor. The second method, suitable even for users with little or no experience in programming or APWIN Basic, is via the LEARN mode (Procedure Learn Mode menu command). Starting LEARN mode causes each ensuing user mouse click and keyboard entry to write a line of APWIN Basic code into the Procedure Editor. Simple procedures may be completely generated in LEARN mode. More sophisticated procedures with branching, calling of sub-procedures, processing of data results, etc., can have their core created in LEARN mode but will typically require further commands to be added in the Macro Editor.

The Learn Mode Toolbar  contains icons to start or stop Learn Mode. When Learn Mode is activated, operator actions including the result of mouse clicks, menu selections, and text or numeric entries into panel fields, will result in lines of APWIN Basic language code being automatically written into the Macro Editor. The resulting macro can then be run to re-create the series of actions.

LEARN mode is started with by clicking on the Learn icon  on the Learn Mode toolbar, or by selecting from the menus the Procedure and Learn Mode or Utilities and Learn Mode selections. Once Learn mode has been started, user actions will result in one or more lines of program code written into the Macro Editor until Learn mode is halted. If a macro has already been loaded into the Macro Editor, the commands created by Learn mode will be inserted at the cursor position in the Macro Editor. If no procedure has been loaded, the Macro Editor will be opened with a new (blank) macro ready for recording of the Learn mode commands. To stop Learn mode, click on the Stop Learning icon  or use the Procedure Learn Mode or Utilities Learn Mode menu selections again to toggle Learn Mode off. To temporarily suspend the learning of commands, hold down the 

and SHIFT keys while clicking the mouse to make changes which will not be learned.

For a Learn Mode example, assume the following list of user actions:

- Click on Start Learn Mode icon.
- Click on New Test icon.
- Click on analog generator On/Off control.
- Click on analog analyzer Ch A input and select Gen Monitor instead of XLR Bal.
- Click on page 2 tab.
- Click on GO (or press F9).
- Click on Stop Learn Mode icon.

Opening the Macro Editor should show the program listing as illustrated in Figure 3-1. This procedure will duplicate all the actions above if the Run Procedure icon is clicked.

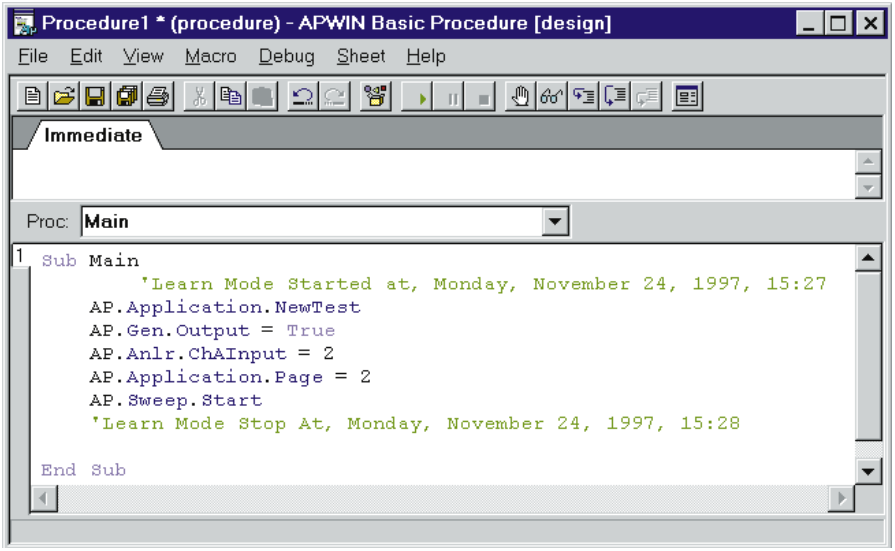


Figure 3-1

Example APWIN Basic Program

```
' This program is designed to assist in creating limit
' files for FFT tests.  It Is intended to be executed
' after a test has already been setup and run.
'
' Functionally, this program will take the results of
' a sweep and limit the low amplitude data points to a
'  specific value.  This is particularly useful for
' limit files based on FFT sweeps where the low
' amplitude data is often near the noise floor and
' varies from sweep to sweep.
'
' Algorithmically, the program operates by
' transferring the sweep data into an array in APWIN
' Basic.  This array is scaled from linear units into
' decibels.  Each data point in the array is tested
' against a specific limit and if the data is above the
' limit it is left untouched.  If it is equal to or
' below the limit, it is forced equal to the limit.
' Once all the data has been processed it is
' transferred back to APWIN and redisplayed.  A limit
' file can then created from this data.

Const Ch1_limit = -110     ' units for limit are in dB
Const Ch2_limit = -110     ' units for limit are in dB

Sub Main
   Call scale_low_amplitudes
End Sub

Sub scale_low_amplitudes
   size = AP.Sweep.Source1.Steps + 1 ' determine number _
      of elements in data arrays
   data1 = AP.Data.XferToArray(0, 1)
   data2 = AP.Data.XferToArray(0, 2)
```

```
   For i = 0 To size          ‘ convert data to dB format
      data1(i) = TodB(data1(i))
      data2(i) = TodB(data2(i))
   Next i

   For i = 0 To size    ‘ limit minimum values to -110 dB
      If data1(i) < Ch1_limit Then
         data1(i) = Ch1_limit
      End If
      If data2(i) < Ch2_limit Then
         data2(i) = Ch2_limit
      End If
   Next i

   For i = 0 To size          ‘ convert data back from dB
      data1(i) = ToExp(data1(i))
      data2(i) = ToExp(data2(i))
   Next i

   For i = 0 To size             ‘ write data back to AP
      AP.Data.Value(0,1,i) = data1(i)
      AP.Data.Value(0,2,i) = data2(i)
   Next i
   AP.Data.UpdateDisplay(0)’Show updated results on graph
End Sub

Function TodB(x)
   TodB = 20*Log10(x)
End Function ‘TodB

Function ToExp(x)
   ToExp = Exp10(x/20)
End Function ‘ToExp
```

Program Structure

APWIN Basic programs can be broken down into three main sections:

- a header section
- the Main sub procedure
- additional sub procedures and functions

The header section of a program can contain several different parts. Any variables, constants, arrays, and other data types that must be accessible to other code modules should be declared in the header section. The amount of program code in the header section can vary significantly depending on whether the macro is self contained, or includes other code modules and public variables. You will learn more about how and where to define variables later in this chapter.

A second and often neglected use of the header section is for comments. A good macro header should have a few sentences that identify who wrote the macro, when it was written, what the macro does, and maybe a few words about how it works. Taking the time to add comments to the header section will help you to quickly identify what your macro does and how it works months or even years later when you need to make a change. A more thorough discussion of how and when to use comments is covered in the next section.

Experienced Visual Basic programmers may recognize that it isn't strictly necessary to have a header section for a macro. If you have developed a very simple macro that doesn't use public variables or include other code modules, it is possible to have the first line of your program begin with the Sub Main declaration. While this minimalist approach will work, it tends to lead to code that is poorly commented and should be avoided.

The Sub Main procedure was introduced in chapter 1. Its purpose is to identify where program execution begins and every APWIN Basic program must include a Sub Main procedure to run. Depending on the complexity of your program, you may only need this one procedure. More typically, however, the main sub procedure is used

as the "top" level of the program from which other sub procedures and functions are called.

Procedures can be listed in your macro in any order you choose. Consider placing the Sub Main procedure as the first procedure in your macro to help others quickly identify where the program starts. Also, if you are using the main sub procedure as the "top" level of your macro, placing it at the start of the program code will help others to quickly identify the how your program flows through the various sub procedures and functions.

After the Sub Main procedure, you should place the additional sub procedures and functions used in your program. Again, there are some tricks you can use to help keep your program as understandable as possible. Structure the sub procedures and functions so that they roughly follow the same order as they are used. In complex programs where the same sub procedures may be called several different times it may not be possible to follow this rigorously. Your goal in structuring your code should be to keep it as simple and easy to understand as you can make it.

Commenting Code

Properly commented code is an essential part of good programming technique. Code which is not properly documented can be hard to read and difficult to modify. In this section we look briefly at some of the reasons to comment your code as well as some useful guidelines.

One of the biggest temptations to resist when developing code, is neglecting to take the time to comment a procedure you just developed for fear you will loose your train of thought or fall behind schedule. This is usually a mistake. Very few programmers possess the discipline to return to their code when it is finished and add the proper comments. Even worse, after you've been away from your code for a while, it may be difficult to remember how everything works. You may not even remember the reasons why you chose one particular way to implement your code over another.

There are several good reasons to add comments to your code. Among the most compelling are:

• Properly commented code will enable you to quickly identify what a procedure does without having to read through the code.

• Comments can help to identify what types of arguments and what ranges of values can be passed to a procedure. This will help you to determine where your code can be re-used.

• Comments are the best chance another programmer has for understanding your code. Code which is not commented or commented poorly is often overlooked by other programmers regardless of how well the code may work. If someone else can't easily understand how your code works, they won't use it.

Some of the goals you should work towards when commenting code include:

• Include general comments about a procedure that allow other programs to quickly and easily identify what the procedures does.

• Identify what input arguments your procedure accepts and what outputs it produces. You should also identify any non-local variables that are used or changed.

• Avoid comments that explain what each line of code does. Anyone who understands APWIN Basic will be able to tell that. What programmers want to see are comments on why your code works the way it does. For example, a `For...Next` loop that counts from one to the number of data points minus one doesn't need a comment saying how many points are counted. What is needed are comments saying why you count up to the number of data points minus one and not *all* the data points.

Commenting code may seem like an added burden that will slow down code development, but any experienced programmer will tell you that well documented code goes a long way towards developing bug free and re-usable code.

Keywords and Commands

At the beginning of this chapter there is an example of an APWIN Basic program. If you study this program, you will notice that there are several keywords and commands that are used to tell APWIN Basic what to do. For example, notice the `If...Then` command used at several points in the code. This command, and others like it, are easily identified in the editor by the different color text. The APWIN Basic editor automatically changes the color of keywords and commands as they are entered. You'll find this coloring scheme makes it much easier to read the code and identify the keywords and commands that control program operation.

A careful observer may have also noticed that none of the variable or constant names are the same as any of the keywords or commands. This is because keywords are reserved in APWIN Basic. If you try to create a variable with the name *end*, APWIN Basic will recognize *end* as one of its keywords. When you try to run a program with a variable named *end* APWIN Basic will refuse to continue and issue an error message.

For an overview of the different keywords available in APWIN Basic, select the APWIN Basic Language option under the Help menu in APWIN.

APWIN Basic offers a large number of keywords and commands to provide you flexibility in creating programs. In the next few sections we will study more closely how to use these to create your own APWIN Basic programs.

Using Variables and Constants

As you develop an APWIN Basic program, you will often need to store information in your program, even if only temporarily. For example, you might need to calculate a running sum of data and you want to be able to store this value while your code loops through all the data. APWIN Basic, like other programming languages, uses *variables* for storing information. Depending on the type of variables you use, the information stored in a variable may only be available during the short

time in which your procedure uses it, or the information may be preserved during the entire time the program is executed.

A variable stores information which may change as your program is run. In order to use variables, Visual Basic must know something about the type of data the variable will store, known as the *data type*. It must also have a *name*, or label it uses to refer to the value the variable contains.

A *constant* is similar to a variable except its value does not change as the program is executed. You use constants to simplify your code and make it easier to read. Like variables, constants have specific names and data types.

Declaring Variables

Before APWIN Basic can use a variable, that variable must first be *declared*. Declaring a variable means that APWIN Basic reserves a location in memory to store information that is assigned to the variable. The amount of memory reserved depends on the data type used.

Variables can be declared in one of two ways, either *explicitly* or *implicitly*. An explicitly declared variable is created by a specific line of code that identifies the variable name and, optionally, its data type. An implicitly declared variable is not specifically identified in a separate line of code, but is used just as if it had been explicitly declared.

There a several statements used in APWIN Basic to declare variables. The following briefly describes these statements and when they should be used.

Declaration Statement	Description
Dim	Used to declare variables within procedures that disappear from memory when the procedure ends. It can also be used to declare variables shared by all procedures in a program.

Static	Used to declare variables within a procedure that remain in memory when the procedure ends.
Public	Used to declare variables shared by all procedures in a project. A project may contain several different programs.
Private	Used to declare variables available only to procedures in the current module.

Variables declared with the Dim statement follow the general form:

```
Dim VariableName As DataType
```

All other variable types are declared in the same way, by adding the declaration statement before the variable name.

```
Public VariableName As DataType
Private VariableName As DataType
Static VariableName As DataType
```

Note that any variables declared as Public should be placed at the beginning of your program before any sub or function procedures. Public variables cannot be declared within sub procedures.

Scope of Variables

Variables can be created that are accessible to all procedures in a program, or they can be restricted to use only in a specific procedure. How *visible* a variable is to different procedures is known as the scope of the variable. There are three levels of scope:

- Local
- Macro level
- Public

Local variables have the narrowest scope. They are only visible to the procedure where they are declared and used. This means you can have several variables in your program, each with the same name, as long as they are declared locally in separate sub and function procedures.

To ensure a variable is local, declare it either implicitly or explicitly inside a procedure. Here is an example sub procedure with three locally declared variables, two of which are declared explicitly (A1 and A2) and one of which is declared implicitly (A3):

```
Sub DoSomething
  Dim A1 As String
  Static A2 As Integer
  A3 = 4.0
  ...
End Sub
```

Local variables are useful when you need to temporarily store information in a procedure. A local variable declared implicitly or with the `Dim` statement will be removed from memory when the procedure is finished executing. `Static` variables will remain in memory and can be used again the next time the procedure is called. By definition, all local variables are private to the procedure in which they are used.

Macro level variables have a much broader scope than local variables. A macro level variable is visible to all procedures in the module (remember, a macro is the same as a .apb file, and you can link together several different code modules with the `'#uses` command discussed in the previous chapter).

To create a macro level variable it must be declared outside of any sub or function procedures. Typically, you should place these in the header section of your program.

The primary advantage of macro level variables is that they can be used to easily share information between different procedures. When one procedure assigns a specific value to a macro level variable, a second procedure can access and use that same information.

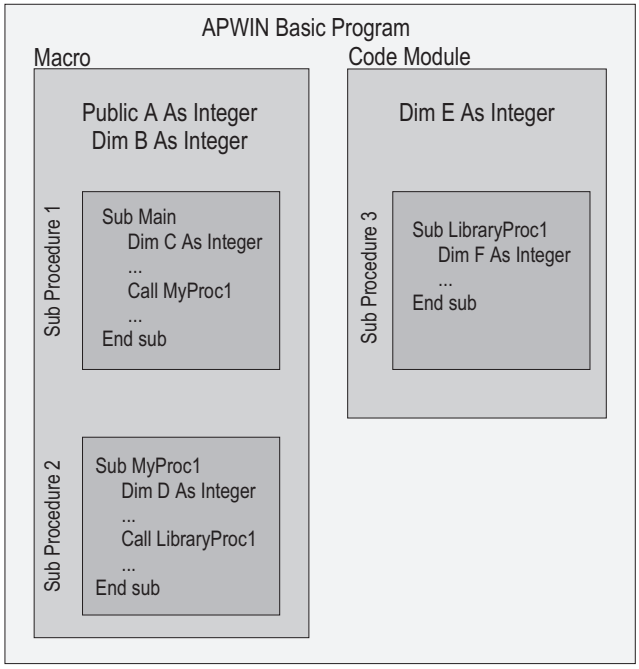
Public variables have the broadest scope and are visible to all sub and function procedures in an application, regardless of the module that contains them. They are declared using the Public statement and should be placed at the top of a module prior to the first procedure. Here is a simple example of declaring and using a Public variable.

```
Public Y As Integer

Sub Main
  Y = 1
  Y = Y + 10
  ...
End Sub
```

Figure 3-2 shows how the scope and visibility of variables change depending on how and where they are declared.

When APWIN Basic is executing code, it evaluates variables starting from the narrowest scope to the broadest. Therefore, if your code contains a local variable, a module level variable, and a `public` variable each with the same name, APWIN Basic will look first for a local variable with the desired name, then for the module level variable, and finally, it will check for a `public` variable.



Procedure	Variable visible to procedure
1	A, B, C
2	A, B, D
3	A, E, F

Figure 3-2

Data Types

When you declare a variable, you can optionally supply a data type. A *data type* is a property that identifies what type of data is stored in a variable. The particular data type a variable assumes specifies two things:

- the type of data (i.e. text, numeric, object)
- the range of values for the data

The following table describes a few of the more common data types available in APWIN Basic.

Data Type	Storage Size	Range
Integer	2 bytes	-32,768 to 32,767
Single	4 bytes	±3.4 E38 to ±1.4 E-45
String	1 byte per character	0 to approximately 65,500 characters
Boolean	2 bytes	**True** or **False**
Variant	16 bytes + 1 byte	depends on data type assumed for each character.

You can learn more about all of the available data types in the online help.

The Variant Data Type

The variant data type is a special data type. By default, any variable that is not explicitly assigned a data type will be assumed to be variant. It is the most flexible data type available in APWIN Basic since it can assume the value of any other data type. The particular data type a variant assumes depends on how the variable is used. For example, a variable with the variant data type can be assigned an integer value at the start of a program, and then be reassigned to a string value later in the code. It changes data types depending on how it is used. Consider the following example:

```
Dim FFTSize              ‘ Variant data type by default
FFTSize = “1024"         ‘ FFTSize is a string data type
FFTSize = FFTSize * 8    ‘ FFTSize changes to a  numeric
‘  data type equal    to 8192
FFTSize = “Big” & FFTSize  ‘ FFTSize is now a string
                            ‘ again containing “Big8192"
```

Constants

A *constant* is a name you choose to replace a value used in your program. They are used to help make code both easier to read and to modify.

For example, suppose you need to use the value of Pi = 3.145926535 at several different places in your code. You could type in the value of Pi each time you need it, but this takes time and is prone to error. Instead, using a constant with the name Pi will be faster and easier to read. Later in your code if you determine you wanted to use 2*Pi instead, you only need to change the value of the constant.

You declare constants with the Const statement:

```
Const name = value
```

Here is how to use Pi as a constant :

```
Const Pi = 3.145926535
```

You don't need to declare the data type for a constant because APWIN Basic simply determines the data type based on its value. For the example shown above, Pi is assigned the *double* data type.

Controlling Program Flow

In this section you will learn how to write procedures that can test conditions and run certain branches of code depending upon the results. The APWIN Basic commands that make decisions and alter code flow are called *control structures*. A second class of commands known as *loop structures* can be used to execute the same section of code multiple times.

Earlier, when introducing procedures it was said that code is executed in a procedure from top to bottom, one line at a time. Although simple procedures can be written using such linear flow, much of the power and utility of APWIN Basic comes from its ability to use control structures to change the order in which code is run.

The diagram in figure 3-3 illustrates the three most common types of program control flow.

Control Structures

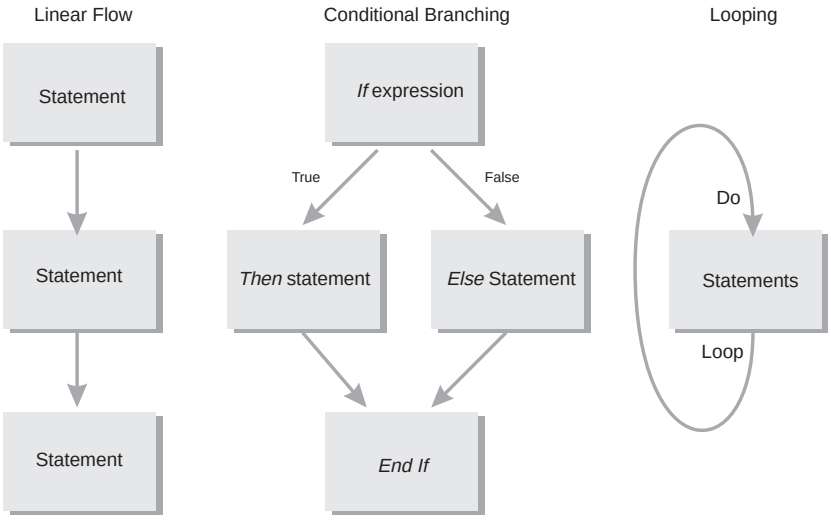


Figure 3-3

If...Then

The *If...Then* structure is used to run a section of code depending on the evaluation of a test expression. The test expression must be either true or false. When the expression is true, the section of code inside the If...Then structure is run. If the expression is false, the code is skipped.

You can use either a single-line syntax or a multiple-line syntax.

```
If FFTSize 2048 Then MsgBox “Use a larger FFT Size”
```

- OR -

```
If FFTSize < 2048 Then
    MsgBox “Use a larger FFT Size”
End If
```

Notice that the multiple-line syntax uses the `End If` statement to identify where the code section ends. If you want to run more than one line of code when the condition is true, you must use the multiple-line syntax.

```
If FFTSize < 2048 Then
    FFTSize = 2048
    MsgBox “FFT Size has been increased to 2048"
End If
```

If...Then...Else

This is a more flexible form of the `If...Then` structure. It allows you define more than one section of code, one of which is always run.

```
If Age 18 Then
    MsgBox “You are too young to vote.”
Else
    MsgBox “You are old enough to vote.”
End If
```

You can add the `Else If` statement within the `If..Then` structure for even more flexibility.

```
If Season = “Summer” Then
    Temperature = “hot”
Else If Season = “Spring” Or “Fall” Then
    Temperature = “mild”
Else
    Temperature = “cold”
End If
```

Notice that last possible season, “Winter” was not tested with an `Else If` statement. If the season is neither summer, spring, or fall, then it must be winter. It is possible to use the `Else If` statement to test for winter, but you would get the same result.

Select Case

APWIN Basic provides the `Select Case` statement as an alternative to `If...Then...ElseIf`. The select case statements searches for matching values to an expression instead of testing whether the expression is true or false. Often, it is used to make code more efficient and readable.

```
Select Case Percentile
   Case Is > 50
      MsgBox "Above the 50th percentile"
   Case 50
      MsgBox "perfectly average"
   Case Else
      MsgBox "Below the 50th percentile"
End Select
```

Notice the use of the `Is` operator to compare a range of values to the initial expression.

The first line of code in a select case statement identifies the expression to be evaluated. For the example just given, the expression is Percentile. The select case statement can be used to evaluate only one expression, unlike the `If...Then...Else` structure which can test several different, even unrelated, expressions.

Loop Structures

For...Next

The `For...Next` structure is used to loop through a section of code a specific number of times. It uses a variable to count the number of times the loop has been run. Depending on how you want the code to run, the variable is incremented or decremented on each loop through the code. Execution stops when the variable reaches a predetermined value.

```
For y = 1 To 10
   MsgBox "The count is currently " & CStr (y)
Next y
```

In this example, `y` is the count variable. It is initialized to 1 at the start of the loop and is incremented on each pass. A message box indicates the value of the `y`. When `y` is equal to 10 a final message is given and the loop terminates.

You can make the `For...Next` structure more flexible by counting either up or down and by using a variable step size.

```
For i = 16 To 4 Step -2
   MsgBox "The count is currently " & CStr (i)
Next i
```

This example will count down from 16 to 4 by steps of two.

Do...Loop

The Do...Loop structure is used to count an indeterminate number of times. Instead of a count variable, it uses a test expression to determine when execution should stop. In this way, a Do...Loop structure will run until the expression is satisfied.

```
Sub IncrementByTwo (x)
   Dim LimitReached As Boolean
   LimitReached = False
   loopCount = 0
   Do Until LimitReached
      x = x + 2
      If x > 100 Then
         MsgBox “The limit was reached in ” & _
            CStr(loopCount) & “ loops”
         LimitReached = True
      Else
         loopCount = loopCount + 1
      End If
   Loop
End Sub
```

This sub procedure accepts an unknown input x from the calling procedure. It then increments the value of x by two until x is greater than 100. When the test condition is satisfied the boolean expression LimitReached is changed from false to true and a message is given reporting the number of times the loop was run.

An alternate way to use the Do...Loop structure is use the Do While clause instead of the Do Until clause. If you use the Until clause, the loop runs as long as the expression is false. When you use the While clause the loop runs as long as the expression is true. Its important that the code in a Do...Loop structure provides a means to alter the test expression. If the test expression can't change, APWIN Basic will not be able to exit the loop.

User Notes

User Notes

User Notes

User Notes

User Notes

Testing and Debugging APWIN Basic Code

Once you have written an APWIN Basic application, you need to determine if your application runs properly. This is part of testing your code. If it does not run correctly, you need a means to fix these errors, also known as debugging your code. APWIN Basic cannot diagnose or fix errors for you, but it does provide a number of tools to help you analyze how your code operates.

APWIN Basic uses an Interactive Design Environment (IDE) to assist in detecting and fixing errors in your program. In this environment it is possible to stop your code at any point during execution and display the state of variables and properties. You can also step through your code one line at a time while watching how settings change. The ability to interact with your code as it is executing is a powerful debugging tool.

Unfortunately, there are no magic tricks to debugging, and there are no steps that always catch errors. Debugging is really part of a process to help you better understand how your code is operating. Using the debugging tools provided in the Interactive Design Environment it is possible to more easily identify and correct the problems that keep your application from running properly.

Different Types of Programming Errors

Before exploring how to test and debug code, consider the kinds of errors you might encounter.

- *Syntax errors* occur when code is improperly written. For example, incorrectly typing a keyword, using incorrect punctuation, and omitting key words are all forms of syntax errors. APWIN Basic will detect and flag these errors before the code is run.

- *Run-time errors* result when a section of code is impossible to execute. A common example you may have encountered before is a divide by zero error. These types of errors cannot be detected until the code is executed. When APWIN Basic encounters a run-time error, program execution is halted.

• *Logic errors* are the most common and can be one of the most difficult types of errors to fix. A logic error occurs when code doesn't operate the way it was intended. Even though the code may be syntactically correct and will run without errors, it may not produce the results you expect. APWIN Basic cannot detect logic errors since it can't know how your program should work. It does, however, provide a number of tools to help you diagnose logic errors.

As you first develop your code, you're likely to create a number of syntax errors. These are easy to detect since APWIN will point them out to you by highlighting the affected line in red and placing the cursor close to the suspected error when you run the macro. As you become more proficient in APWIN Basic, you will tend to make fewer syntax errors.

Once your program is syntactically correct, you can execute it. At this point, you may or may not encounter run-time errors. These errors often occur only for certain types of input data, so you may or may not see them the first time your program runs. In fact, you may have to run your code several different times and with several different sets of data before you see a run-time error.

Lastly, you may notice logic errors when your program runs but behaves differently than you expected. Any of these three types of errors will require you to review your code, identify the source of the bug, and re-write your code to fix the error.

Using the Debugging Tools on the Toolbar

The APWIN Basic editor has a number of buttons used for debugging code. These buttons are found near the top of the Procedure Editor panel.

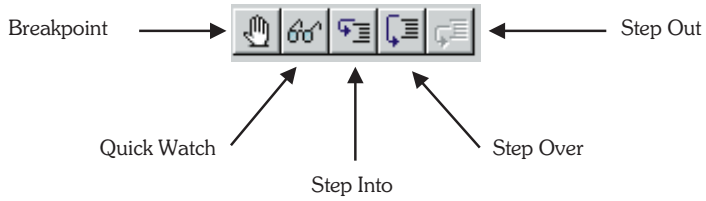


Figure 4-1

The following table describes the function of each button

Debugging Tool	Purpose
Breakpoint	Used to mark a line in the code where Visual Basic will suspend execution.
Quick Watch	Displays the value of the expression under the cursor while in break mode.
Step Into	Executes the next line of code in the application and steps into procedures.
Step Over	Executes the next line of code in the application without stepping into procedures.
Step Out	Steps out of the current sub procedure or function.

These debugging tools are designed to help you observe the behavior of your code and enable you to diagnose and fix run-time and logic errors. In order to use these tools effectively, you need to understand how they can be utilized during program operation.

Break Mode

Break mode is a special operating mode of APWIN Basic that allows you to halt program execution and examine the state of variables and expressions in your code. When you enter break mode:

- The Debug window automatically appears in the procedure editor panel as shown in figure 4-3. The Debug window includes several different window panes that provide useful debugging information.
- You are temporarily prevented from editing your code. Since you have actually just suspended execution but not stopped execution, APWIN Basic does not allow you to add and remove commands from your program.

Once you have entered break mode, the value of all variables and expressions is preserved, so you can check their current state. Depending on whether or not your program is running correctly, you may want to change the value of several variables and expressions as well. In break mode it is possible to interact with program operation in several ways.

While in break mode you can:

- Check the value of variables, expressions, and properties.
- Modify the value of variables and expressions.
- Use the immediate pane in the Debug window to run APWIN Basic commands not included in your program.
- Step through operation of your code one line or one procedure at a time.

Accessing Break Mode

APWIN Basic will enter break mode when any of the following occur:

- Execution reaches a line of code with a breakpoint.
- Execution reaches a *Stop* statement.
- A line of code generates a run-time error.
- Program execution is started by pressing either the Step Into, Step Over, or Step Out buttons. 

The most common technique for accessing break mode is to add breakpoints to your code. APWIN Basic will enter break mode and suspend execution on the line of code just before the breakpoint.

To add a breakpoint, move the cursor to the line of code where you want to place a breakpoint and press the toggle breakpoint  icon. When you set a breakpoint, APWIN Basic will mark the selected line of code by adding a small dot at the start of the line as shown in figure 4-2. To remove a breakpoint, select the desired line of code and press the toggle breakpoint button.

A second way of entering break mode is to add the *Stop* command to your code. This is most useful when you need to ensure program execution halts at a particular point. Notice, there is an important difference between breakpoints and the Stop command. Breakpoints are lost when you close and reload your program, but Stop statements stay in the code until you remove them.

Regardless of how you entered break mode, you can always resume execution by pressing the run/resume button  or by continuing to step through your code.

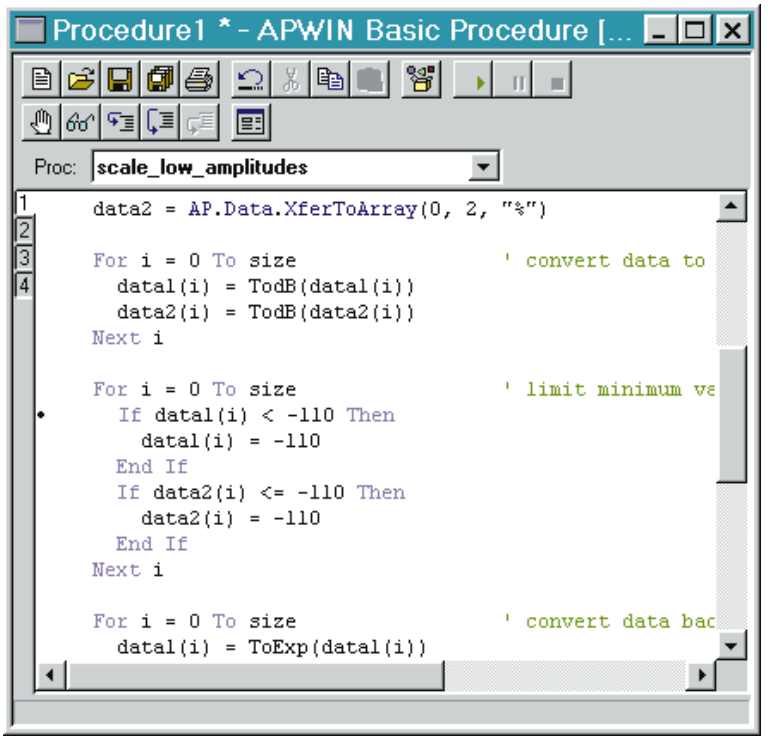


Figure 4-2

Stepping Through Code

Once you've identified a potential trouble spot in your code, it is useful to continue executing your code one line at a time. This allows you to see how each line affects the behavior of the application as well as the values of variables and other data. Executing code one line at a time is called *stepping through code*. APWIN Basic provides three different tools to step through your code.

- Step Into
- Step Over
- Step Out

These three tools operate nearly the same. When you press any of them, APWIN Basic will execute the next line of code and then return to break mode. They differ in how they execute a line of code that either calls another procedure or that exists inside of a called procedure.

For example, if the current line of code to be executed is a call to another procedure, Step Into will move into that next procedure. Step Over, on the other hand, will not descend into the called sub procedure. Instead, it executes all the commands in the called sub procedure and halts immediately after returning to the calling procedure. This is useful if you are reasonably certain that the bug you're looking for isn't in the called procedure and you don't want to take the time to step through it.

Step Out will execute all the commands in the current procedure until it has returned to the calling procedure. Once it has reached the calling procedure it halts execution and returns to Break Mode. You should use Step Out if you have stepped through all the code in the current procedure you are interested in and you want to return to the calling procedure. Note, if you press Step Out from the Main sub procedure, and you have not added any additional breakpoints to your code, the program will run to completion.

Using The Debug Window

In the Debug window, you can monitor the values of expression and variables while stepping through the statements in your code. There are four window panes available in the Debug window, the Immediate, Watch, Stack, and Loaded. Each of these window panes can provide useful debugging information about your program.

You display the debug window by:

- Entering Break Mode. The Debug window is automatically opened when APWIN Basic enters Break Mode.

- Choosing **View** and then **Always Split** from the menu options available when you right-click the mouse in the main editor window.

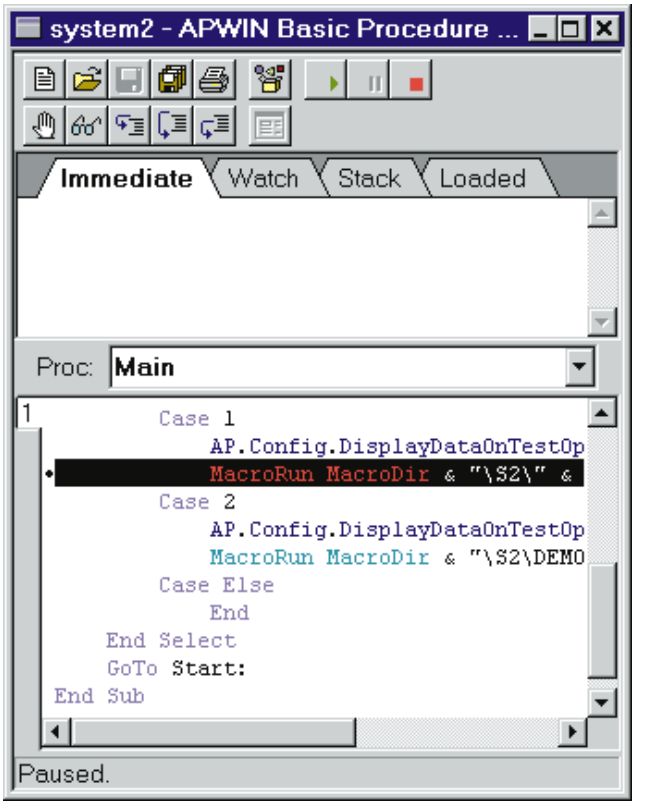


Figure 4-3

This will leave the Debug window visible in the Procedure Editor panel as shown in figure 4-3.

The *Watch pane* displays information about expressions and variables you tell APWIN to monitor as your code is executing. The *Immediate pane* allows you to enter additional APWIN Basic commands to learn more about your code. Typically, you use the Immediate pane to change the value of a variable or expression. The *Stack pane* shows you information about what line of code is currently active and what procedures have been called to reach the current line. Finally, the *Loaded pane* indicates all the .apb files that have been loaded and are being used by the current program.

Additional information about all of the window panes shown in the Debug window is available in the online help.

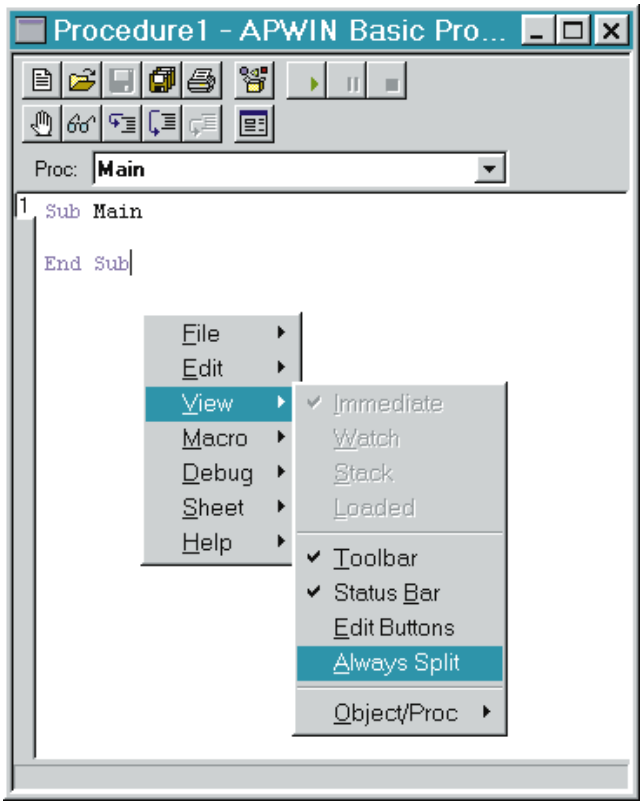


Figure 4-4

Error Handling

In addition to testing and debugging your code, it is valuable to consider the different ways you can develop code to handle errors that occur while your program is running. When a run-time error occurs, APWIN Basic will usually generate an error message that halts your code. Often, there's nothing the user can do to resume running the application. Other errors might not interrupt execution, but they may cause it to act unpredictably. From a programmers standpoint, it's important to know how to write code that can detect run-time errors and branch to special code that will recover from the errors without

halting your program. Adding code to recover from errors is known as *error handling*.

There are several different ways run-time errors can be generated. Earlier, when discussing the different types of errors, it was mentioned that code attempting a divide by zero will generate a run-time error. More generally, a run-time error occurs whenever your code attempts an invalid instruction. For example, you might have a procedure that prompts the user to enter the name of a test file to run. If the user enters an invalid name or a name that does not exist, APWIN Basic will not be able to continue. In this section, we consider different techniques you can use to recover from run-time errors.

APWIN Basic Error Handling Commands

APWIN Basic provides a number of commands to allow you to detect and handle run-time errors before they halt your program (a program that abruptly halts operation and won't continue is said to have *crashed*). Intercepting an error is also known as *trapping* an error. You can use the following statements to trap and then respond to run-time errors:

- The **On Error Goto** command can be used to branch in your code when an error is detected. It must be set up before the run-time error occurs.

- The **Err** function returns the number corresponding to the most recent run-time error.

- The **Error** function returns message text corresponding to an error number. Every run-time error has a corresponding error number that identifies it.

The following example uses all three types of error handing commands.

```
Sub Main
   X = 1
   Y = 0
   On Error GoTo ErrorMessage
      Z = X/Y  ‘ create a divide by zero error
      ‘ At this point the code moves to the _
          ErrorMessage section

   Exit Sub    ‘ leave the procedure at this point

ErrorMessage:
   MsgBox “The most recent error number is ” _
   & Err & “.  The error message is: ” & Error(Err)
   Resume Next
   ‘ return to next line of code after the error occurred
End Sub
```

When you run this program, it will generate a message box that says, “The most recent error number is 10061. The error message is: Divide by zero.”

Notice that this example has introduced several new programming techniques. The first technique to consider is the use of the **Goto** command. Whenever the Goto command is used, it must refer to a *line label* in your program. In the preceding example, the line label used in the Goto command was “ErrorMessage:” All line labels must follow the standard APWIN Basic naming conventions and must end with a colon.

The second technique to notice is the use of the line continuation command. This is the underscore character “_”, seen at the end of the line beginning with the MsgBox command. The line continuation command tells APWIN Basic to wrap the next line of code into the current line of code.

Lastly, the **Resume Next** command is used to return from error branching. It allows your program to continue normal operation after handling the error condition.

The process of trapping errors can be summarized as:

- Setting an error trap
- Writing code to handle to the error
- Returning to normal program execution

User Notes

User Notes

Creating Custom User Interfaces

Many of the macros you are likely to develop in APWIN Basic will be designed to assist in automating tests and simplifying complex measurements. One of the most powerful ways to simplify using a macro is to include a custom user interface. You create a custom user interface by adding code that will create dialog boxes and custom menus when your macro is executed.

A custom user interface can be very useful when you want to guide a novice APWIN user through running a number of different tests. For example, a macro might begin by presenting the user with a custom menu that offers several different tests to run. Different tests can be linked to different menu options depending on the type of measurement needed. The user can only select from the tests available. When a chosen test is complete, the results can be printed out or logged to a file and the macro then returns to the initial custom menu as shown in figure 5-1.

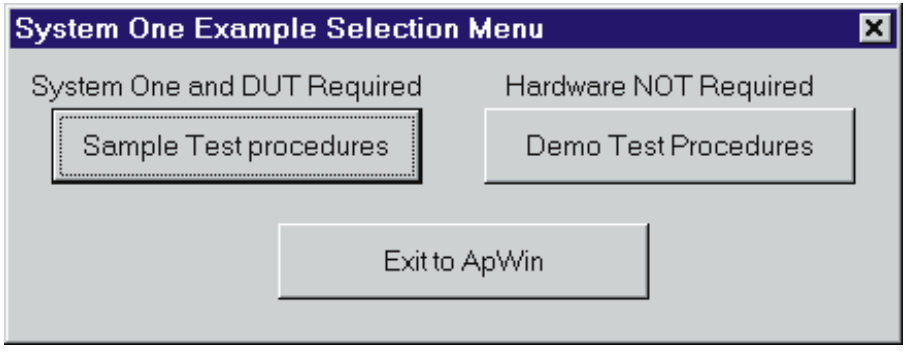


Figure 5-1

This section explains how to use dialog boxes and menus to customize the user interface to your macros. Among the different tasks you can complete with dialog boxes and menus include:

• Getting information from the user. A typical example might include querying the user for their initials which can be logged in the test report.

• Displaying information to the user. Message boxes can be developed indicating how the hardware should be connected or what errors may have occurred while testing.

• Simplifying the interface of APWIN with custom menus. With a properly constructed custom interface, a user does not need to be familiar with the subtleties of APWIN.

To assist in developing custom dialog boxes and menus, APWIN Basic includes a dialog box editor showh in figure 5-2. To access the dialog box editor press the button on the macro editor panel. This will bring up a generic template for a dialog box. You can select from the menu bar on the left of the dialog box editor to define regions of text in your message box as well as locations for push button controls or user input. Figure 5-3 shows a previously created dialog box that has been highlighted. Once highlighted the  button is pressed to edit the dialog box.

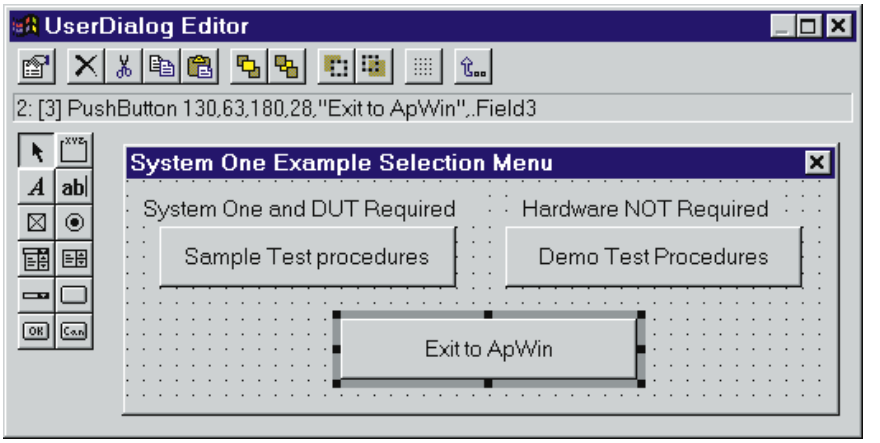


Figure 5-2

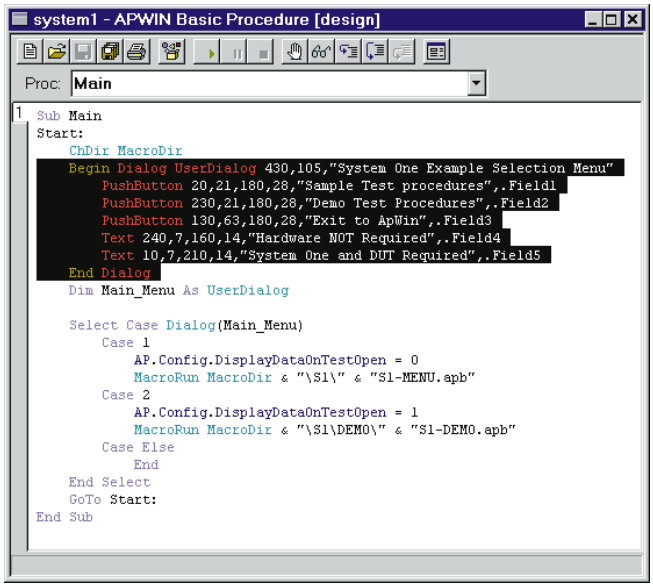


Figure 5-3

An example of implementing a custom user interface is shown on the following page. Notice that when the macro is run, the code will remain in a loop waiting for the user to select a menu option. When a particular option is selected, the Macro Run command is used to launch a second macro that executes the desired test. When complete, the macro will close and return to the main loop.

```
Sub Main
Start:
   ChDir MacroDir
   Begin Dialog UserDialog 430,105,"System One Example _
      Selection Menu"
      PushButton 20,21,180,28,"Sample Test _
         procedures",.Field1
      PushButton 230,21,180,28,"Demo Test Procedures", _
         .Field2
      PushButton 130,63,180,28,"Exit to ApWin",.Field3
      Text 240,7,160,14,"Hardware NOT Required",.Field4
      Text 10,7,210,14,"System One and DUT Required", _
         .Field5
   End Dialog
   Dim Main_Menu As UserDialog

   Select Case Dialog(Main_Menu)
      Case 1
          AP.Config.DisplayDataOnTestOpen = 0
         MacroRun MacroDir & "\S1\" & "S1-MENU.apb"
      Case 2
          AP.Config.DisplayDataOnTestOpen = 1
         MacroRun MacroDir & "\S1\DEMO\" & "S1-DEMO.apb"
      Case Else
         End
   End Select
   GoTo Start:
End Sub
```

User Notes

User Notes

Reports

As you become more familiar with APWIN and APWIN Basic, you might want to begin sharing information with other applications. For example, you may want to create a report document as shown in figure 6-1 that lists the data results gathered from several test procedures. Or you might want to move the sweep results from an APWIN test into Excel to determine the RMS solution. Whatever your needs, APWIN Basic can be used to automate the exchange of information with other applications.

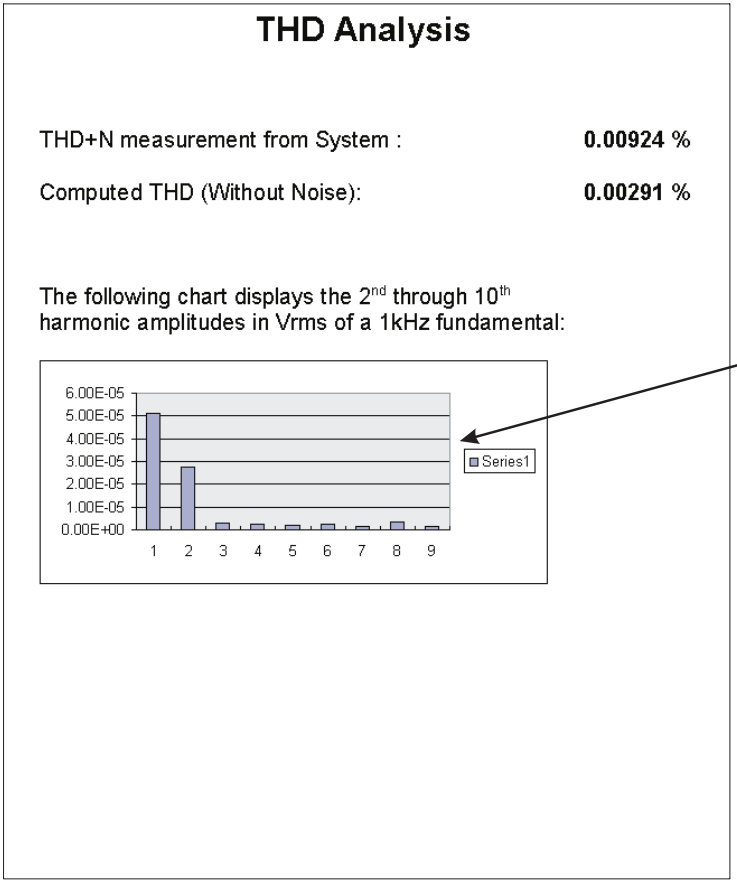


Figure 6-1

APWIN Basic works with other applications through *OLE Automation*. Applications that support OLE Automation provide *objects* that can be used in your programs. Some of the things you can do with other applications that support OLE Automation include:

- Creating, opening or saving documents
- Adding, deleting or retrieving data including text and charts
- Executing commands and procedures in APWIN Basic that alter how the external application operates.

The topic of OLE Automation and how to use it to control other Applications is well documented in a number of different books specializing in OLE Automation. Future editions of this manual will include more information on the mechanics of using OLE Automation in APWIN Basic.

The rest of this chapter includes an example of OLE Automation a typical APWIN user might be interested in. In this example, results from an APWIN test are moved into Excel shown in figure 6-2 where the RMS of the data is computed. Next, the same information is

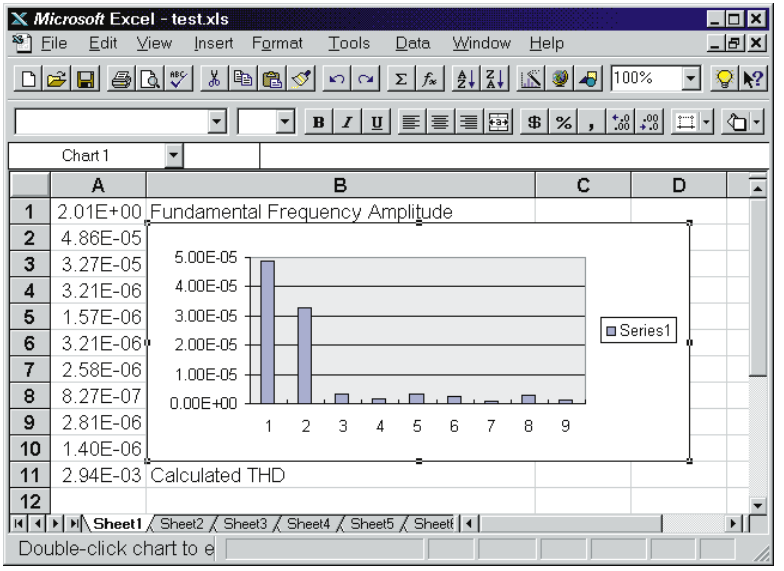


Figure 6-2

inserted into an already existing Word document shown in figure 6-3

at several key locations. The updated Word document is then printed. The advantage of using OLE Automation is that the entire process is automated, and all the code to complete the task is centralized in one APWIN Basic macro.

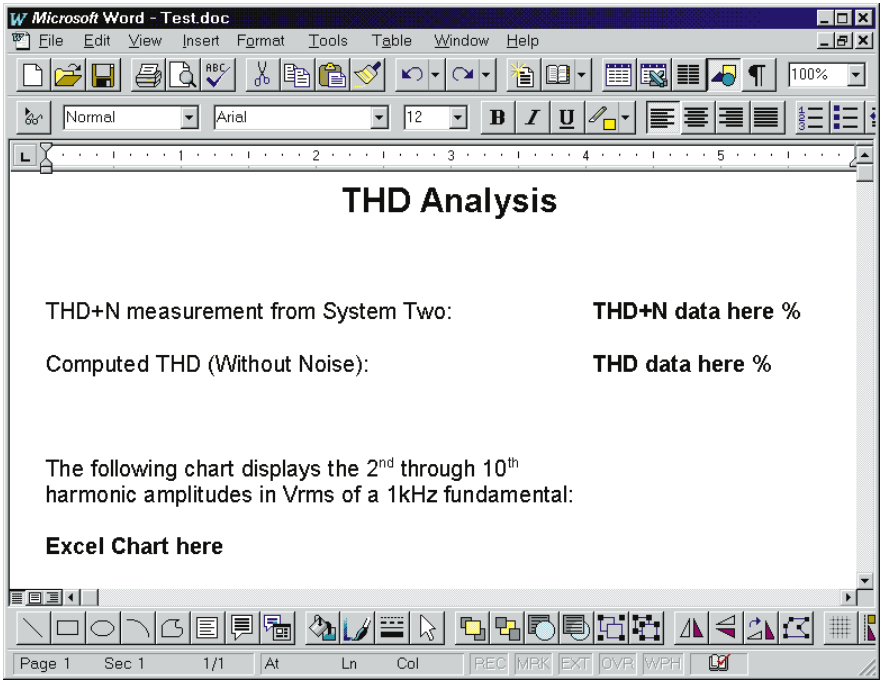


Figure 6-3

Example

```
' OLE example program
' created 10/02/96 by William Rich
' last modified 10/02/96

' This is an example procedure demonstrating the use
' of OLE 2.0 in APWIN Basic.  OLE, or Object Linking
' and Embedding, can be used to share information
' between different OLE compliant applications.  In
' this example, data is shared between three OLE
' applications, APWIN, Microsoft Excel, and Microsoft
' Word.
'
' The procedure performs the following three tasks.
' Task 1 involves loading and running a previously
' created APWIN test for System Two. Task 2 takes the
' results from the APWIN test and imports them into an
' existing Excel spreadsheet.  Using Excel, the RMS of
' the data is computed and a chart is created.
' Finally, Task 3 takes the results from both APWIN
' and Excel and inserts them into a Word document to
' create a report.

Dim dataCh1() As Double 'Holds measurements from the
                         '  APWIN sweep test
Dim Rms                 'RMS of harmonics calculated
                         '  in Excel spreadsheet
Dim Thd                 'Holds calculated THD value
Dim numPoints           'Number of data points used
                         '  during sweep
Dim reading             'Stores THD+N measurement from
                         '  analog analyzer
Dim LevelA              'Stores Level measurement from
                         '  analog analyzer


Sub Main
   Call GetDataFromAPWIN
   Call ProcessDataInExcel
   Call SaveResultsInWord
End Sub
```

```
Sub GetDataFromAPWIN()
   AP.File.OpenTest “THD.AT2"
   AP.Sweep.Start

   numPoints = AP.Data.ColSize(0, 1) ‘ Number of  data _
      points in sweep
   dataCh1 = AP.Data.XferToArray(0, 1, “V”)’ Transfer _
      sweep results to dataCh1 array - data is in Volts

   AP.Anlr.ChALevelTrig
   While AP.Anlr.ChALevelReady      ‘ Wait for settled _
      Level measurement
   Wend
   LevelA = AP.Anlr.ChALevelRdg(“V”)    ‘ Get Level _
      reading in Volts

   AP.Anlr.FuncTrig
   While AP.Anlr.FuncReady           ‘ Wait for settled _
      THD+N measurement
   Wend
   reading = AP.Anlr.FuncRdg(“%”) ‘ Get THD+N reading _
      in percent
   reading = CStr(Format$(reading, “0.00000"))
‘Convert reading variable from a double to a string
‘  for use in Word document
End Sub

Sub ProcessDataInExcel()
   Dim MSExcel As Object
   Dim chart As Object

‘Start a copy of Excel
   Set MSExcel = CreateObject(“Excel.Application”)
‘Excel is normaly invisible on startup. Set to visible
   MSExcel.Application.Visible = True
   MSExcel.Workbooks.Open _
      Filename:="C:\APWIN\OLE\TEST.XLS"

‘Insert fundamental frequency amplitude into cell #1
   MSExcel.Cells(1, 1).Value = LevelA
```

```
   For i = 1 To numPoints     'Insert data into Excel
      MSExcel.Cells(i + 1, 1).Value = dataCh1(i - 1)
   Next i

   MSExcel.Cells(numPoints + 2, 1).Formula = _
      "=(SQRT(SUMSQ(A2:A10))/A1)*100"

   Thd = CStr(Format$(MSExcel.Cells(11, 1),"0.00000"))

   MSExcel.Range("A2:A10").Select 'Select input data _
      as active
   MSExcel.Selection.Copy 'Copy data to insert in chart

'Create a chart of the data we just added.  Assign
'  the Excel chart to the "chart" object.
   Set Chart = MSExcel.ActiveSheet.ChartObjects.Add _
      (49, 13, 271.5, 119.25)
   Chart.Select         'Select chart as active
   Chart.Copy           'Copy chart to clipboard

   MSExcel.Workbooks("TEST.XLS").Close saveChanges:=False
   MSExcel.Application.Quit
End Sub

Sub SaveResultsInWord()
   Dim MSWord As Object
   Set MSWord = CreateObject("Word.Basic") 'Start Word
'Word is normaly invisible on startup.  Set to visible
   MSWord.AppShow
   MSWord.FileOpen Name:="C:\APWIN\OLE\TEST.DOC"

'Search for string
   MSWord.EditFind "THD+N data here"
'Paste THD+N reading from APWIN
   If MSWord.EditFindFound Then MSWord.Insert reading
'Search for string
   MSWord.EditFind "THD data here"
'Paste THD computation
   If MSWord.EditFindFound Then MSWord.Insert Thd
'Search for string
   MSWord.EditFind "Excel Chart here"
```

```
'Paste Excel chart from clipboard
   If MSWord.EditFindFound Then MSWord.EditPaste

   MSWord.FilePrint           'Print Doc from MS Word
   MSWord.FileCloseAll 2      'Close all open files
   MSWord.AppClose            'Close MS Word
End Sub
```

User Notes

Language Reference

Introduction

Groups

Declaration	#Reference, #Uses, Attribute, Class Module, Code Module, Const, Declare, Deftype, Dim, Enum...End Enum, Function...End Function, Object Module, Option, Private, Property...End Property, Public, ReDim, Static, Sub...End Sub, Type...End Type, WithEvents.
Assignment	Erase, Let, LSet, RSet, Set.
Flow Control	Call, Do...Loop, End, Exit, For...Next, For Each...Next, GoTo, If...ElseIf...Else...EndIf, MacroDir, MacroRun, MacroRunThis, Select Case...End Case, Stop, While...Wend,
Error Handling	Err, Error, On Error, Resume.
Conversion	Array, CBool, CByte, CCur, CDate, CDbl, CInt, CLng, CSng, CStr, CVar, CVDate, CVErr, Val.
Variable Info	IsArray, IsDate, IsEmpty, IsError, IsMissing, IsNull, IsNumeric, IsObject, LBound, TypeName, UBound, VarType.
Math	Abs, Atn, Cos, dBToPowerRatio, dBToVoltageRatio, Exp, Exp10, Fix, Int, Log, Log10, Pow, PowerRatioTodB, Randomize, Rnd, Sgn, Sin, Sqr, Tan, VoltageRatioTodB.
String	Asc, AscB, AscW, Chr, ChrB, ChrW, Format, Hex, InStr, InStrB, InStrRev, LCase, Left, LeftB, Len, LenB, LTrim, Mid, MidB, Oct, Replace, Right, RightB, RTrim, Space, String, Str, StrComp, StrConv, Trim, UCase.
Object	CreateObject, GetObject. With...End With.
Time/Date	Date, DateAdd, DateDiff, DatePart, DateSerial, DateValue, Day, Hour, Minute, Month, Now, Second, Time, Timer, TimeSerial, TimeValue, Weekday, Year.
File	ChDir, ChDrive, Close, CurDir, Dir, EOF, FileAttr, FileCopy, FileDateTime, FileLen, FreeFile, Get, GetAttr, Input, Input, Kill, Line

	Input, Loc, Lock, LOF, MkDir, Name, Open, Print, Put, Reset, RmDir, Seek, Seek, SetAttr, Unlock, Write.
User Input	Dialog, GetFilePath, InputBox, MsgBox.
User Dialog	Begin Dialog...End Dialog, CancelButton, CheckBox, ComboBox, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, Picture, PushButton, Text, TextBox.
Dialog Function	Dialog Func, DlgControlId, DlgCount, DlgEnable, DlgEnd, DlgFocus, DlgListBoxArray, DlgName, DlgNumber, DlgSetPicture, DlgText, DlgType, DlgValue, DlgVisible.
DDE	DDEExecute, DDEInitiate, DDEPoke, DDERequest, DDETerminate, DDETerminateAll.
Settings:	DeleteSetting, GetAllSettings, GetSetting, SaveSetting
Miscellaneous	AppActivate, Attribute, Beep, CallersLine, Choose, Clipboard, Command, Debug.Print, DoEvents, Environ, IIf, MacroDir, QBColor, Rem, RGB, SendKeys, Shell, Wait, WaitAndDoEvents.
Operator	Operators: +, -, ^, *, /, \, Mod, +, -, &, =, <>, <, >, <=, >=, Like, Not, And, Or, Xor, Eqv, Imp, Is.

Operators

Syntax

```
^ Not * / \ Mod + - & < <= > >= = <> Is And Or Xor
Eqv Imp
```

Description

These operators are available for numbers n1 and n2 or strings s1 and s2. If any value in an expression is *Null* then the expressions value is *Null*. The order of operator evaluation is controlled by operator *precedence*.

Operator	Description
-n1	Negate *n1*.
n1 ^ n2	Raise *n1* to the power of *n2*.
*n1 * n2*	Multiply *n1* by *n2*.
n1 / n2	Divide *n1* by *n2*.
n1 \ n2	Divide the integer value of *n1* by the integer value of *n2*.
n1 Mod n2	Remainder of the integer value of *n1* after dividing by the integer value of *n2*.
n1 + n2	Add *n1* to *n2*.
s1 + s2	Concatenate s1 with *s2*.
n1 - n2	Difference of *n1* and *n2*.
s1 & s2	Concatenate s1 with *s2*.
n1 < n2	Return *True* if *n1* is less than *n2*.
n1 <= n2	Return *True* if *n1* is less than or equal to *n2*.
n1 > n2	Return *True* if *n1* is greater than *n2*.
n1 >= n2	Return *True* if *n1* is greater than or equal to *n2*.
n1 = n2	Return *True* if *n1* is equal to *n2*.
n1 <> n2	Return *True* if *n1* is not equal to *n2*.
s1 < s2	Return *True* if s1 is less than *s2*.
s1 <= s2	Return *True* if s1 is less than or equal to *s2*.
s1 > s2	Return *True* if s1 is greater than *s2*. \
s1 >= s2	Return *True* if s1 is greater than or equal to *s2*.
s1 = s2	Return *True* if s1 is equal to *s2*.
s1 <> s2	Return *True* if s1 is not equal to *s2*.
Not n1	Bitwise invert the integer value of *n1*. Only Not *True* is *False*.
n1 And n2	Bitwise and the integer value of *n1* with the integer value *n2*.
n1 Or n2	Bitwise or the integer value of *n1* with the integer value *n2*.

n1 Xor n2	Bitwise exclusive-or the integer value of *n1* with the integer value *n2*.
n1 Eqv n2	Bitwise equivalence the integer value of *n1* with the integer value *n2* (same as Not (n1 Xor n2)).
n1 Imp n2	Bitwise implicate the integer value of *n1* with the integer value *n2* (same as (Not n1) Or n2).

Example

```
Sub Main
    N1 = 10
    N2 = 3
    S1$ = “asdfg”
    S2$ = “hjkl”
    Debug.Print -N1         ‘-10
    Debug.Print N1 ^ N2     ‘ 1000
    Debug.Print Not N1      ‘-11
    Debug.Print N1 * N2     ‘ 30
    Debug.Print N1 / N2     ‘ 3.3333333333333
    Debug.Print N1 \ N2     ‘ 3
    Debug.Print N1 Mod N2   ‘ 1
    Debug.Print N1 + N2     ‘ 13
    Debug.Print S1$ + S2$   ‘"asdfghjkl"
    Debug.Print N1 - N2     ‘ 7
    Debug.Print N1 & N2     ‘"103"
    Debug.Print N1 < N2     ‘False
    Debug.Print N1 <= N2    ‘False
    Debug.Print N1 > N2     ‘True
    Debug.Print N1 >= N2    ‘True
    Debug.Print N1 = N2     ‘False
    Debug.Print N1 <> N2    ‘True
    Debug.Print S1$ < S2$   ‘True
    Debug.Print S1$ <= S2$  ‘True
    Debug.Print S1$ > S2$   ‘False
    Debug.Print S1$ >= S2$  ‘False
    Debug.Print S1$ = S2$   ‘False
    Debug.Print S1$ <> S2$  ‘True
    Debug.Print N1 And N2   ‘ 2
    Debug.Print N1 Or N2    ‘ 11
    Debug.Print N1 Xor N2   ‘ 9
    Debug.Print N1 Eqv N2   ‘ -10
    Debug.Print N1 Imp N2   ‘ -9
End Sub
```

Data Types

Any, Boolean, Byte, Currency, Date, Double, Integer, Long, Object, Single, String, String*n, Variant, user type.

Type	Description
`Any`	Any variable expression (**Declare** only).
`Boolean`	A *True* or *False* value.
`Byte`	An 8 bit unsigned integer value.
`Currency`	A 64 bit fixed point real. (A twos complement binary value scaled by 10000.)
`Date`	A 64 bit real value. The whole part represents the date, while the fractional part is the time of day. (December 30, 1899 = 0.) Use #date# as a literal date value in a macro.
`Double`	A 64 bit real value.
`Integer`	A 16 bit integer value.
`Long`	A 32 bit integer value.
`Object`	An object reference value. (see Objects)
`PortInt`	A portable integer value. For Win16: A 16 bit integer value. For Win32: A 32 bit integer value.
`Single`	A 32 bit real value.
`String`	An arbitrary length string value.
`String*n`	A fixed length (n) string value.
`UserDialog`	A *usertype* defined by **Begin Dialog** UserDialog.
`Variant`	An empty, numeric, currency, date, string, object, error code, null or array value.

Keywords

Empty, False, Nothing, Null, True. Win16, Win32.

Word	Description
Empty	A *variantvar* that does not have any value.
False	A *condexpr* is false when its value is zero. A function that returns False returns the value 0.
Nothing	An *objexpr* that does not refer to any object.
Null	An *variant expression* that is null. A null value propagates through an expression causing the entire expression to be Null. Attempting to use a Null value as a string or numeric argument causes a run-time error. A Null value prints as #NULL#.

Example

```
Sub Main
    X = Null
    Debug.Print X = Null '(even this expression is _
Null)
    Debug.Print IsNull(X) '(use IsNull to test for a _
Null value)
End Sub
```

Example Output

```
Null
True
```

True	A *conditional expression* is true when its value is non-zero. A function that returns *True* returns the value -1.
Win16	**True** if running in 16 bits. **False** if running in 32 bits.
Win32	**True** if running in 32 bits. **False** if running in 16 bits.

Language Commands

Abs — Function

Syntax `Abs(num)`

Parameters

Name	Description
num	Return the absolute value of this number value.

Description Return the absolute value.

Example

```
Sub Main
   Debug.Print Abs(9)
   Debug.Print Abs(0)
   Debug.Print Abs(-9)
End Sub
```

Example Output

```
 9
 0
 9
```

AppActivate — Instruction

Syntax `AppActivate title$`

-or-

`AppActivate TaskID`

Parameters

Name	Description
title$	The name shown in the title bar of the window.
TaskID	This numeric value is the task identifier.

Description Form 1: Activate the application top-level window titled Title$. If no window by that title exists then the first window with at title that starts with Title$ is activated. If no window matches then an error occurs.

Form 2: Activate the application top-level window for task TaskID. If no window for that task exists then an error occurs.

See Also SendKeys, Shell( ).

Example

```
Sub Main
   'Make ProgMan the active application
   AppActivate "Program Manager"
End Sub
```

Array — Function

Syntax **Array(**[*expr*[, ...]]**)**

Description Return a variant value array containing *exprs*.

Example

```
Sub Main
   X = Array(0,1,4,9)
   Debug.Print X(2)
End Sub
```

Example Output

```
4
```

Asc — Function

Syntax **Asc(***string$***)**

Parameters

Name	Description
string$	Return the ASCII value of the first char in this string value.

Description Return the ASCII value.

Note: A similar function, AscB, returns the first byte in S$. Another similar function, AscW, returns the Unicode number.

See Also Chr$( ).

Example

```
Sub Main
   Debug.Print Asc("A")
End Sub
```

Example Output

```
65
```

Atn

Function

Syntax `Atn(num)`

Parameters

Name	Description
num	Return the arc tangent of this number value. This is the number of radians. There are 2*Pi radians in a full circle.

Description Return the arc tangent.

Example

```
Sub Main
   Debug.Print Atn(1)*4
End Sub
```

Example Output

```
3.14159265358979
```

Attribute

Definintion/Statement

Syntax `Attribute name = value`

Description All attribute definitions and statements are ignored except for:

o

Public varname As Type

Attribute varname.VB_VarUserMemId = 0

Declares Public varname as the default property for a class module or object module.

o

Property [Get|Let|Set] propname (...)

Attribute propname.VB_UserMemId = 0

...

End Property

Declares Property propname as the default property for a class module or object module.

Beep

Instruction

Syntax **Beep**

Description Sound the bell.

Example

```
Sub Main
   Beep          'Beep the bell.
End Sub
```

Begin Dialog

Definition

Syntax

```
Begin Dialog UserDialog [x, y,] dx, dy[, title$][,
.dialogfunc]
  User Dialog Item
  [User Dialog Item]...
End Dialog
```

Parameters

Name	Description
x	This number value is the distance from the left edge of the screen to the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font. If this is omitted then the dialog will be centered.
y	This number value is the distance from the top edge of the screen to the top edge of the dialog box. It is measured in 1/12ths of the average character width for the dialogs font. If this is omitted then the dialog will be centered.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
title$	This string value is the title of the user dialog. If this is omitted then there is no title.
dialogfunc	This is the function name that implements the **DialogFunc** for this UserDialog. If this is omitted then the *UserDialog* doesnt have a dialogfunc.

User Dialog Item — One of: CancelButton, CheckBox, ComboBox, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox.

Description Define a *UserDialog type* to be used later in a **Dim** As *UserDialog* statement.

See Also `Dim As UserDialog.`

Example

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button."
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg  show dialog (Wait for OK)
End Sub
```

Call — Instruction

Syntax

```
Call name[(arglist)]
-or-
name[arglist]
```

Description Evaluate the *arglist* and call subroutine (or function) *name* with those values. Sub (or function) *name* must be previously defined by either a **Sub** (or **Function)** definition. If *name* is a function then the result is discarded. If Call is omitted then *name* must be a subroutine and the *arglist* is not enclosed in parens.

See Also **`Declare, Sub.`**

Example

```
Sub Show(Title$,Value)
   Debug.Print Title$;" =";Value
End Sub
Sub Main
   Call Show("2000/9",2000/9)
   Show "1",1<2          'True
End Sub
```

Example Output

```
222.2222222222
True
```

CallersLine — Function

Syntax CallersLine[(Depth)]

Description Return the caller's line as a text string.

The text format is: "[macroname|subname#linenum] linetext".

Parameter	Description
Depth	This integer value indicates how deep into the stack to get the caller's line. If Depth = 0 then return the current line. If Depth = 1 then return the calling subroutine's current line, etc.. If Depth is greater than the call stack then a null string is returned. If this value is omitted then the depth is 1.

Example

```
Sub Main
    A
End Sub
Sub A
    Debug.Print CallersLine ' "[(untitled 1)|Main#  2]
A"
End Sub
```

CancelButton Dialog Item — Definition

Syntax **CancelButton** *x*, *y*, *dx*, *dy*[, *.field*]

Parameters

Name	Description
x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.

y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
field	This identifier is the name of the field. The dialogfunc receives this name as string. If this is omitted then the field name is Cancel.

Description Define a cancel button item. Pressing the Cancel button from a **Dialog** instruction causes a run-time error. (**Dialog**( ) function call returns 0.)

See Also `Begin Dialog, Dim As UserDialog.`

Example

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,30,"Please push the Cancel button"
      OKButton 40,90,40,20
      CancelButton 110,90,60,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg  show dialog (wait for cancel)
   Debug.Print “Cancel was not pressed”
End Sub
```

CBool — Function

Syntax `CBool(num|$)`

Parameters

Name	Description
num	Any number.
$	The string must be either a number in quotes, or True or False in quotes (not case sensitive).

Description Convert to a boolean value. Zero converts to *False*, while all other values convert to *True*.

Example

```
Sub Main
   Debug.Print CBool(-1)
   Debug.Print CBool(0)
   Debug.Print CBool(1)
End Sub
```

Example Output

```
True
False
True
```

CByte — Function

Syntax `Byte(num|$)`

Parameters

Name	Description
num|$	Convert a number or string value to a byte value.

Description Convert to a byte value.

Example

```
Sub Main
   Debug.Print CByte(1.6)
End Sub
```

Example Output

```
2
```

CCur — Function

Syntax `CCur(num|$)`

Parameters

Name	Description
num|$	Convert a number or string value to a currency value.

Description Convert to a *currency* value.

Example

```
Sub Main
   Debug.Print CCur(1E6)
End Sub
```

Example Output

```
 1000000
```

CDate — Function

Syntax `CDate(num|$)`
-or-
`CVDate(num|$)`

Parameters

Name	Description
num|$	Convert a number or string value to a date value.

Description Convert to a *date* value.

Example

```
Sub Main
   Debug.Print CDate(2)
End Sub
```

Example Output

```
1/1/00
```

CDbl — Function

Syntax `CDbl(num|$)`

Parameters

Name	Description
num|$	Convert a number or string value to a double precision real.

Description Convert to a *double* precision real.

Example

```
Sub Main
   Debug.Print CDbl("1E6")
End Sub
```

Example Output

```
 1000000
```

ChDir — Instruction

Syntax `ChDir name$`

Parameters

Name	Description
name$	This string value is the path and name of the directory.

Description Change the current directory to *Name$*.

See Also `ChDrive, CurDir$( ).`

Example

```
Sub Main
   ChDir "C:\"
   Debug.Print CurDir$()
End Sub
```

Example Output

```
C:\
```

ChDrive — Instruction

Syntax `ChDrive drive$`

Parameters

Name	Description
drive$	This string value is the drive letter.

Description Change the current drive to *dfrive$*.

See Also `ChDir, CurDir$( ).`

Example

```
Sub Main
    ChDrive "B"
    Debug.Print CurDir$()
```

Example Output

```
B:\
```

CheckBox Dialog Item — Definition

Syntax `CheckBox x, y, dx, dy, title$, .field`

Parameters

Name	Description
x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.
y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
field	The value of the check box is accessed via this field. Checked is 1, and unchecked is 0.

Description Define a checkbox item.

See Also `Begin Dialog, Dim As UserDialog.`

Example

```
Sub Main
```

```
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button."
      CheckBox 10,25,180,15,"&Checkbox",.Check
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   dlg.Check = 1
   Dialog dlg           'Show dialog (wait for OK)
   Debug.Print dlg.Check
End Sub
```

Example Output

```
0
```

or

```
1
```

Choose — Function

Syntax **Choose(***index, expr*[, ...])

Parameters

Name	Description
index	The numeric value indicates which *expr* to return. If this value is less than one or greater than the number of *exprs* then *Null* is returned.
expr	All expressions are evaluated.

Description Return the value of the *expr* indicated by *Index*.

See Also If, Select Case, IIf( ).

Example

```
Sub Main
   Debug.Print Choose(2,"Hi","there")
End Sub
```

Example Output

```
there
```

Chr$ Function

Syntax `Chr[$](num)`

Parameters

Name	Description
num	Return one char string for this ASCII number value.

Description Return a one char string for the ASCII value.

Note: A similar function, ChrB, returns a single byte ASCII string. Another similar function, ChrW, returns a single char Unicode string.

See Also `Asc( ).`

Example

```
Sub Main
   Debug.Print Chr$(48)
End Sub
```

Example Output

```
0
```

CInt Function

Syntax `CInt(num|$)`

Parameters

Name	Description
num|*$*	Convert a number or string value to a 16 bit integer.

Description Convert to a 16 bit *integer*. If *num*|*$* is too big (or too small) to fit then an overflow error occurs.

Example

```
Sub Main
   Debug.Print CInt(1.6)
End Sub
```

Example Output

```
 2
```

Class

Module

Description: (The Class module feature is not implemented in version 1.5 of APWIN Basic)

A class module implements an OLE Automation object.

o Has a set of Public properties, functions and subroutines accessible from other macros and modules.

o These public symbols are accessed via an object variable.

o Public Consts, Types, arrays, fixed length strings are not allowed.

o A class module is similar to a object module except that no instance is automatically created.

o To create an instance use:

```
Dim Obj As classname
Set Obj = New classname
```

See Also Code Module, Object Module, Uses.

Example

```
'A.WWB
'#Uses "File.CLS"
Sub Main
    Dim File As New File
    File.Attach "C:\AUTOEXEC.BAT"
    Debug.Print File.ReadLine
End Sub

'File.CLS
'File|New Module|Class Module
'Edit|Properties|Name=File
Option Explicit
Dim FN As Integer
Public Sub Attach(FileName As String)
    FN = FreeFile
    Open FileName For Input As #FN
End Sub
Public Sub Detach()
    If FN <> 0 Then Close #FN
    FN = 0
```

```
End Sub
Public Function ReadLine() As String
    Line Input #FN,ReadLine
End Function

Private Sub Class_Initialize()
    Debug.Print “Class_Initialize”
End Sub

Private Sub Class_Terminate()
    Debug.Print “Class_Terminate”
    Detach
End Sub
```

Class_Initialize Sub

Syntax

```
Private Sub Class_Initialize()
    ...
End Sub
```

Description Class module initialization subroutine. Each time a new instance is created for a class module the Class_Initialize sub is called. If Class_Initialize is not defined then no special initialization occurs.

See Also Code Module, Class_Terminate.

Class_Terminate Sub

Syntax

```
Private Sub Class_Terminate()
    ...
End Sub
```

Description Class module termination subroutine. Each time an instance is destroyed for a class module the Class_Terminate sub is called. If Class_Terminate is not defined then no special termination occurs.

See Also Code Module, Class_Initialize.

Clipboard — Instruction/Function

Syntax

```
Clipboard text$
-or-
Clipboard[$][( )]
```

Parameters

Name	Description
text$	Put this string value into the clipboard.

Description Form 1: Set the clipboard to Text$. This is like the Edit|Copy menu command.

Form 2: Return the text in the clipboard.

Example

```
Sub Main
   Debug.Print Clipboard$()
   Clipboard "Hello"
   Debug.Print Clipboard$()
End Sub
```

Example Output

```
Hello
```

CLng — Function

Syntax `CLng(num|$)`

Parameters

Name	Description
num|$	Convert a number or string value to a 32 bit integer.

Description Convert to a 32 bit *long* integer. If *num|$* is too big (or too small) to fit then an overflow error occurs.

Example

```
Sub Main
   Debug.Print CLng(1.6)
End Sub
```

Example Output

```
 2
```

Close

Instruction

Syntax `Close [[#]streamnum][, ...]`

Parameters

Name	Description
streamnum	Streams 1, 2, 3 and 4 are available in each macro. If this is omitted then all open streams for the current macro are closed.

Description Close *streamnums.*

See Also `Open, Reset.`

Example

```
Sub Main
   'Read the first line of XXX and print it.
   Open "C:\APWIN\SAMPLES\SYSTEM1.APB" For Input As #1
   Line Input #1,L$
   Debug.Print L$
   Close #1
End Sub
```

Example Output

```
Sub Main
```

Code Module

Description (The Code module feature is not implemented in version 1.5 of APWIN Basic). A Code module implements a code library.

· Has a set of Public properties, functions and subroutines accessible from other macros and modules.

· The public symbols are accessed directly.

See Also `Class Module, Object Module, Uses.`

ComboBox Dialog Item

Definition

Syntax **ComboBox** *x, y, dx, dy, strarray$*( ), *.field$*

Parameters

Name	Description
x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.
y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
strarray$( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
field$	The value of the combo box is accessed via this field. This is the text in the edit box .

Description

Define a combobox item. Combo boxes combine the functionality of an edit box and a list box.

See Also

`Begin Dialog, Dim As` *`UserDialog.`*

Example

```
Sub Main
Dim combos$(3)
   combos$(0) = “Combo 0"
   combos$(1) = “Combo 1"
   combos$(2) = “Combo 2"
   combos$(3) = “Combo 3"
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      ComboBox 10,25,180,60,combos$(),.combo$
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   dlg.combo$ = none
   Dialog dlg        ‘show Dialog (Wait For ok)
   Debug.Print dlg.combo$
End Sub
```

Example Output

```
Combo 0
```

or

```
Combo 1
```

or

```
Combo 2
```

or

```
Combo 3
```

Command$ Function

Syntax `Command[$]`

Description Contains the value of the **MacroRun** parameters.

See Also MacroRun

Example

```
Sub Main              'Macro 1 Calling Macro.
   MacroRun "MACRO2.APB 1,2,3"
End Sub
Sub Main              'Macro 2 "MACRO2.APB"
   Debug.Print "Command line parameter is: ";
   Debug.Print Command$;
End Sub
```

Example Output

```
Command line parameter is: 1,2,3
```

Const Definition

Syntax `[Private|Public] Const name[type] [As type] = expr[, ...]`

Description Define *name* as the value of *expr*. The *expr* may refer to other constants or built-in functions. If the type of the constants is not specified, the type of *expr* is used. Constants defined outside a **Sub**, **Function** or **Property** block are available in the entire macro.

Private is assumed if neither *Private* or *Public* is specified.

Note: Const statement in a **Sub**, **Function** or **Property** block may not use *Private* or *Public*.

Example

```
Sub Main
Const Pi = 4*Atn(1), e = Exp(1)
   Debug.Print Pi
   Debug.Print e
End Sub
```

Example Output

```
 3.14159265358979
 2.71828182845905
```

Cos — Function

Syntax `Cos(num)`

Parameters	Name	Description
	num	Return the cosine of this number value. This is the number of radians. There are 2*Pi radians in a full circle.

Description Return the cosine.

Example

```
Sub Main
   Debug.Print Cos(1)
End Sub
```

Example Output

```
 0.54030230586814
```

CreateObject — Function

Syntax `CreateObject(class$)`

Parameters	Name	Description
	class$	This string value is the applications registered class name. If this application is not currently active it will be started.

Description Create a new object of type *Class$*. Use **Set** to assign the returned object to an object variable.

See Also Objects

Example

```
Sub Main
   Dim Excel As Object
```

```
   Set Excel = CreateObject("Excel.Application")
   With Excel
      Excel.Visible = True
      Excel.Quit
   End With
   Set Excel = Nothing
End Sub
```

CSng — Function

Syntax `CSng(num|$)`

Parameters

Name	Description
num|$	Convert a number or string value to a single precision real.

Description Convert to a *single* precision real. If *num|$* is too big (or too small) to fit then an overflow error occurs.

Example

```
Sub Main
   Debug.Print CSng(Sqr(2))
End Sub
```

Example Output

```
 1.414214
```

CStr — Function

Syntax `CStr(num|$)`

Parameters

Name	Description
num|$	Convert a number or string value to a string value.

Description Convert to a string.

Example

```
Sub Main
   Debug.Print CStr(Sqr(2))
End Sub
```

Example Output

```
1.4142135623731
```

CurDir$ Function

Syntax **CurDir**[$]**(**[*drive$*]**)**

Parameters

Name	Description
drive$	This string value is the drive letter. If this is omitted or null then return the current directory for the current drive.

Description Return the current directory for *Drive$*.

See Also ChDir, ChDrive.

Example

```
Sub Main
   Debug.Print CurDir$()
End Sub
```

Example Output

```
C:\
```

CVar — Function

Syntax `CVar(num|$)`

Parameters

Name	Description
num|$	Convert a number or string value (or object reference) to a variant value.

Description Convert to a *variant value*.

Example

```
Sub Main
   Debug.Print CVar(Sqr(2))
End Sub
```

Example Output

```
 1.4142135623731
```

CVErr — Function

Syntax `CVErr(num|$)`

Parameters

Name	Description
num|$	Convert a number or string value to an error code.

Description Convert to a *variant* that contains an error code. An error code cant be used in expressions.

See Also `IsError.`

Example

```
Sub Main
   Debug.Print CVErr(1)
End Sub
```

Example Output

```
Error 1
```

Date — Function

Syntax `Date[$]`

Description Return todays *date* as a date value.

See Also `Now, Time, Timer.`

Example

```
Sub Main
   Debug.Print Date
End Sub
```

Example Output `2/8/96`

DateAdd — Function

Syntax **`DateAdd(`***`inteval, number, dateexpr`***`)`**

Description Return a date value a number of intervals from another date.

Parameter	Description
interval	This string value indicates which kind of interval to add.
number	Add this many intervals. Use a negative value to get an earlier date.
dateexpr	Calculate the new date relative to this date value. If this value is Null then Null is returned.

Interval	Description
yyyy	Year
q	Quarter
m	Month
d	Day
w	Weekday
ww	Week
h	Hour
m	Minute
s	Second

See Also DateDiff, DatePart.

Example

```
Sub Main
    Debug.Print DateAdd("yyyy",1,#1/1/2000#) '1/1/2001
End Sub
```

DateDiff — Function

Syntax

DateDiff(*inteval, dateexpr1, dateexpr2***)**

Description

Return the number of intervals between two dates.

Parameter	Description
interval	This string value indicates which kind of interval to subtract.
dateexpr1	Calculate the from this date value to dateexpr2. If this value is Null then Null is returned.
dateexpr2	Calculate the from dateexpr1 to this date value. If this value is Null then Null is returned.

Interval	Description
yyyy	Year
q	Quarter
m	Month
d	Day
w	Weekday
ww	Week
h	Hour
m	Minute
s	Second

See Also

DateAdd, DatePart.

Example

```
Sub Main
    Debug.Print DateDiff("yyyy",#1/1/1990#,#1/1/2000#)
' 10
End Sub
```

DatePart — Function

Syntax `DatePart(inteval, dateexpr)`

Description Return the number from the date corresponding to the interval.

Parameter	Description
interval	This string value indicates which kind of interval to extract.
dateexpr	Get the interval from this date value. If this value is Null then Null is returned.

Interval	Description (return value range)
yyyy	Year (100-9999)
q	Quarter (1-4)
m	Month (1-12)
d	Day (1-366)
w	Weekday (1-7)
ww	Week (1-53)
h	Hour (0-23)
m	Minute (0-59)
s	Second (0-59)

See Also DateAdd, DateDiff.

Example

```
Sub Main
    Debug.Print DatePart("yyyy",#1/1/2000#) ' 2000
End Sub
```

DateSerial — Function

Syntax `DateSerial(year, month, day)`

Parameters

Name	Description
year	This numeric value is the year (0 to 9999). (0 to 99 are interpreted as 1900 to 1999.)
month	This numeric value is the month (1 to 12).
day	This numeric value is the day (1 to 31).

Description Return a *date* value.

See Also `DateValue, TimeSerial, TimeValue.`

Example

```
Sub Main
   Debug.Print DateSerial(1996,2,8)
End Sub
```

Example Output

```
2/8/9
```

DateValue — Function

Syntax **`DateValue(`***`date$`***`)`**

Parameters

Name	Description
`date$`	Convert this string value to the day part of date it represents.

Description Return the day part of the date encoded as a string.

See Also `DateSerial, TimeSerial, TimeValue.`

Example

```
Sub Main
   Debug.Print DateValue("2/8/1996 12:00:01 AM")
End Sub
```

Example Output

```
2/8/96
```

Day — Function

Syntax **`Day(`***`dateexpr`***`)`**

Parameters

Name	Description
`dateexpr`	Return the day of the month for this date value.

Description Return the day of the month (1 to 31).

See Also `Date( ), Month( ), Weekday( ), Year( ).`

Example

```
Sub Main
   Debug.Print Day(#1/1/1900#)
```

```
End Sub
```

Example Output 1

dBToPowerRatio — Function

Syntax **dBToPowerRatio(*num*)**

Parameters

Name	Description
num	dB number

Description Return the power ratio of *num* to 1.

Example

```
Sub Main
   Debug.Print Format(dBToPowerRatio(-3), "#.0000")
End Sub
```

Example Output .5012

Equation PowerRatio = Exp10(*num* / 10)

dBToVoltageRatio — Function

Syntax **dBToVoltageRatio(*num*)**

Parameters

Name	Description
num	dB number

Description Return the voltage ratio of *num* to 1.

Example

```
Sub Main
   Debug.Print Format(dBToVoltageRatio(-6), "#.0000")
End Sub
```

Example Output .5012

Equation VoltageRatio = Exp10(*num/20*)

DDEExecute — Instruction

Syntax

```
DDEExecute channum, command$[, timeout]
```

Parameters

Name	Description
channum	This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.
command$	Send this command value to the server application. The interpretation of this value is defined by the server application.
timeout	The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Description

Send the DDE Execute *Command$* string via DDE *Channum*.

Example

```
Sub Main
   ChanNum = DDEInitiate(PROGMAN,"PROGMAN")
   DDEExecute ChanNum,"[CreateGroup(XXX)]"
   DDETerminate ChanNum
End Sub
```

DDEInitiate — Function

Syntax

```
DDEInitiate(app$, topic$)
```

Parameters

Name	Description
app$	Locate this server application.
topic$	This is the server applications topic. The interpretation of this value is defined by the server application.

Description

Initiate a DDE conversation with *App$* using *Topic$*. If the conversation is successfully started then the return value is a channel number that can be used with other DDE instructions and functions.

Example

```
Sub Main
   ChanNum = DDEInitiate (PROGMAN,  PROGMAN)
   DDEExecute ChanNum,"[CreateGroup(XXX)]"
   DDETerminate ChanNum
End Sub
```

DDEPoke — Instruction

Syntax **DDEPoke** *channum*, *item$*, *data$*[, *timeout*]

Parameters

Name	Description
channum	This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.
item$	This is the server applications item. The interpretation of this value is defined by the server application.
data$	Send this data value to the server application. The interpretation of this value is defined by the server application.
timeout	The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Description Poke *Data$* to the *Item$* via DDE *Channum*.

Example

```
Sub Main
   ChanNum = DDEInitiate(PROGMAN,"PROGMAN")
   DDEPoke ChanNum,"Group","XXX"
   progman doesnt support poke
   DDETerminate ChanNum
End Sub
```

DDERequest$ — Function

Syntax **DDERequest[$](***channum*, *item$*[, *timeout*])

Parameters

Name	Description
channum	This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.
item$	This is the server applications item. The interpretation of this value is defined by the server application.
timeout	The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Description Request information for *Item$*. If the request is not satisfied then the return value will be a null string.

Example

```
Sub Main
   ChanNum = DDEInitiate(PROGMAN,"PROGMAN")
   Debug.Print DDERequest$(ChanNum,"Groups")
   DDETerminate ChanNum
End Sub
```

DDETerminate

Instruction

Syntax **DDETerminate** *channum*

Parameters

Name	Description
channum	This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.

Description Terminate DDE *Channum*.

Example

```
Sub Main
   ChanNum = DDEInitiate(PROGMAN,"PROGMAN")
   DDEExecute ChanNum,"[CreateGroup(XXX)]"
   DDETerminate ChanNum
End Sub
```

DDETerminateAll

Instruction

Syntax **DDETerminateAll**

Description Terminate all open DDE channels.

Example

```
Sub Main
   ChanNum = DDEInitiate(PROGMAN,"PROGMAN")
   DDEExecute ChanNum,"[CreateGroup(XXX)]"
   DDETerminateAll
End Sub
```

Debug

Object

Syntax

```
Debug.Print [expr[; ...][;]]
```

Description

Print the *expr*(s) to the output window. Use ; to separate expressions. A *num* is automatically converted to a string before printing (just like **Str$**( )). If the instruction does not end with a ; then a newline is printed at the end.

Example

```
Sub Main
   X = 4
   Debug.Print "X/2 =", X/2
   Debug.Print "Start...";     'Dont Print    a newline
   Debug.Print "Finish"        'Print a newline"
End Sub
```

Example Output

```
X/2 = 2
Start...Finish
```

Declare

Definition

Syntax

```
[Private|Public] Declare Sub name Lib dllname _
[Alias modulename] [([param[, ...]])]
```

-or-

```
[Private|Public] Declare Function name[type] Lib _
dllname [Alias modulename] [([param[, ...]])] As _
type]
```

Parameters

Name	Description
name	This is the name of the subroutine/function being defined.
dll name	This is the DLL file where the modules code is.
module name	This is the name of the module in the DLL file. If this is #number then it is the ordinal number of the module. If it is omitted then name is the module name.

params	A list of zero or more params that are used by the DLL subroutine or function. (Note : A ByVal strings value may be modified by the DLL.)

Description Interface to a DLL defined subroutine or function. The values of the calling *arglist* are assigned to the *params*.

Public is assumed if neither *Private* or *Public* is specified.

WARNING! Be very careful when declaring DLL subroutines or functions. If you make a mistake and declare the paremeters or result incorrectly then Windows might halt. Save any open documents before testing new DLL declarations.

See Also `Function, Sub, Call.`

Example

```
Declare Function GetActiveWindow& Lib "user32" ()
Declare Function GetWindowTextLength% Lib "user32" _
(ByVal hwnd&)
Declare Sub GetWindowText Lib "user32"    (ByVal hwn&%, _
ByVal lpsz$, ByVal   cbMax&)
Function ActiveWindowTitle$()
   ActiveWindow = GetActiveWindow()
   TitleLen =  GetWindowTextLength(ActiveWindow)
   Title$ = Space$(TitleLen)
   GetWindowText  ActiveWindow,Title$,TitleLen+1
   ActiveWindowTitle$ = Title$
End Function
Sub Main
   Debug.Print ActiveWindowTitle$()
End Sub
```

Def — Definition

Syntax **Def**{Bool|Cur|Date|Dbl|Int|Lng|Obj|Sng|Str|Var} *letterrange*[, ...]

Parameters

Name	Description
letterrange	letter, or letter-letter: A letter is one of A to Z. When letter-letter is used, the first letter must be alphabetically before the second letter. Variable names that begin with a

letter in this range default to declared type.
If a variable name begins with a letter not specific in any letterrange then the variable is a Variant. The letterranges are not allowed to overlap.

Description Define untyped variables as:

- DefBool - *Boolean*
- DefByte - *Byte*
- DefCur - *Currency*
- DefDate - *Date*
- DefDbl - *Double*
- DefInt - *Integer*
- DefLng - *Long*
- DefObj - *Object*
- DefSng - *Single*
- DefStr - *String*
- DefVar - *Variant*

See Also `Option Explicit.`

Example

```
DefInt A,C-W,Y          'Integers
DefBool B               'Boolean
DefStr X                'String
                        'All others(Z) are Variant.
Sub Main
   B = 1                'B Is an Boolean.
   Debug.Print B
   X = "A"              'X Is a String.
   Debug.Print X
   Z = 1                'Z Is a Variant (anything).
   Debug.Print Z
   Z = "Z"
   Debug.Print Z
End Sub
```

Example Output

```
 1
A
 1
Z
```

DeleteSetting

Instruction

Syntax

```
DeleteSetting AppName$, Section$[, Key$]
```

Description

Delete the settings for Key in Section in project AppName. Win16 and Win32s store settings in a .ini file named AppName. Win32 stores settings in the registration database.

Parameter	Description
AppName$	This string value is the name of the project which has this Section and Key.
Section$	This string value is the name of the section of the project settings.
Key$	This string value is the name of the key in the section of the project settings. If this is omitted then delete the entire section.

Example

```
Sub Main
    SaveSetting “MyApp”,"Font","Size",10
    DeleteSetting “MyApp”,"Font","Size"
End Sub
```

Dialog

Instruction/Function

Syntax

```
Dialog dialogvar[, default]
-or-
Dialog(dialogvar[, default])
```

Parameters

Name	Description
dlgvar	This variable that holds the values of the fields in a dialog. Use .field to access individual fields in a dialog variable.
default	This numeric value indicates which button is the default button. (Pressing the Enter key on a non-button pushes the default button.) Use -2 to indicate that there is no default button. Other possible values are shown the result table below. If this value is omitted then the first **PushButton**, **OKButton** or **CancelButton** is the default button.

Result

Value	Description
-1	**OK button** was pressed.
0	**Cancel button** was pressed
n	Nth **push button** was pressed.

Description

Display the dialog associated with *dialogvar*. The initial values of the dialog fields are provided by dialogvar. If the **OK button** or any **push button** is pressed then the fields in dialog are copied to the *dialogvar*. The Dialog( ) function returns a value indicating which button was pressed. (See the result table below.)

See Also

`Begin Dialog, Dim As UserDialog.`

Example

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button."
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg        'Show Dialog (Wait For OK)
End Sub
```

DialogFunc

Prototype

Syntax

```
Function Dialogfunc(dlgitem$, action%, suppvalue%)
⇨As Boolean

  Select Case Action%
  Case 1  Dialog box initialization
  ...
  Case 2  Value changing or button pressed
  ...
  Case 3  TextBox or ComboBox text changed
  ...
  Case 4  Focus changed
  ...
  Case 5  Idle
  ...
  End Select
End Function
```

Parameters

Name	Description
dlgitem	This string value is the name of the user dialog items field.
action	This numeric value indicates what action the dialog function is being asked to do.
suppvalue	This numeric value provides additional information for some actions.

Action	Description
1	Dialog box initialization. *DlgItem* is a null string. *SuppValue* is zero.
2	**CheckBox, DropListBox, ListBox** or **OptionGroup**: *DlgItem*s value has changed. *SuppValue* is the new value. **CancelButton, OKButton** or **PushButton**: *DlgItem*s button was pushed. *SuppValue* is meaningless. Set *dialogfunc* = **True** to prevent the dialog from closing.
3	**ComboBox** or **TextBox**: *DlgItem*s text changed and losing focus. *SuppValue* is the number of characters.
4	Item *DlgItem* is gaining focus. *SuppValue* is the item that is losing focus. (The first item is 0, second is 1, etc.)
5	Idle processing. *DlgItem* is a null string. *SuppValue* is zero. Set *dialogfunc* = **True** to continue receiving idle actions.

Description A dialogfunc implements the dynamic dialog capabilities.

See Also Begin Dialog.

Example

```
Sub Main
    Begin Dialog UserDialog 200,120,.DialogFunc
        Text 10,10,180,15,"Please push the OK button."
        TextBox 10,40,180,15,.Text
        OKButton 30,90,60,20
        PushButton 110,90,60,20,"&Hello"
    End Dialog
    Dim dlg As UserDialog
    Debug.Print Dialog(dlg)
End Sub
Function DialogFunc%(DlgItem$, Action%, SuppValue%)
    Debug.Print “Action =”;Action%
    Select Case Action%
    Case 1
        Beep
    Case 2
        If DlgItem$ = “Hello” Then
            MsgBox “Hello”
            DialogFunc% = True ‘do not exit the dialog
        End If
    Case 3
        Debug.Print
DlgItem$;"=""";DlgText$(DlgItem$);"""
    Case 4
        Debug.Print “DlgFocus =”";DlgFocus();"""
    End Select
End Function
```

Dim — Definition

Syntax `Dim name[type][([Dim[,...]])][As type][, ...]`

Description Dimension var array(s) using the *dims* to establish the minimum and maximum index value for each dimension. If the *dims* are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dims*. It must be **ReDim**ensioned before it can be used.

See Also `Begin Dialog, Dialog, Private, Public, ReDim, Static.`

Example

```
Sub DoIt(Size)
   Dim C0,C1(),C2(2,3)
   ReDim C1(Size)       'Dynamic Array
   C0 = 1
   C1(0) = 2
   C2(0,0) = 3
   Debug.Print C0;C1(0);C2(0,0)
End Sub
Sub Main
   DoIt 1
End Sub
```

Example Output

```
 1 2 3
```

Dir$ — Function

Syntax `Dir[$]([pattern$], [attribmask])`

Parameters

Name	Description
pattern$	This string value is the path and name of the file search pattern. If this is omitted then continue scanning with the previous pattern. Each macro has its own independent search. A path relative to the current directory can be used.
attribmask	This numeric value controls which files are found. A file with an attribute that matches will be found.

Description Scan a directory for the first file matching *Pattern$*.

See Also GetAttr( ).

Example

```
Sub Main
   F$ = Dir$("*.*")
   While F$ <> ""
      Debug.Print F$
      F$ = Dir$()
   Wend
End Sub
```

Example Output

```
SNR.APB
FRQ-RESP.AT1
READINGS.APB...
```

DlgControlId

Function

Syntax **DlgControlId(***dlgitem|$***)**

Parameters

Name	Description
dlgitem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog items *field* name.

Description Return the *fields* window id.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Example

```
Sub Main
   Begin Dialog UserDialog 200,120, .DialogFunc
      Text 10,10,180,15,"Please push the OK button."
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
      PushButton 110,90,60,20,"&Hello"
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub
Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action =";Action%
   Select Case Action%
```

```
   Case 1                ‘Dialog box initialization
      Beep
   Case 2                ‘Value changing Or button presse
      If DlgItem$ = Hello Then
         DialogFunc% = True   ‘Do Not Exit the Dialog
      End If
   Case 4                ‘Focused changed
      Debug.Print “DlgFocus = """;DlgFocus();"""
      Debug.Print “DlgControlId(”; DlgItem$;") =";
      Debug.Print DlgControlId(DlgItem$)
   End Select
End Function
```

DlgCount — Function

Syntax `DlgCount()`

Description Return the number of dialog items in the dialog. This instruction/function must be called directly or indirectly from a *dialogfunc*.

Example

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button."
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg
End Sub
Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print “Action =”;Action%
   Select Case Action%
   Case 1            ‘Dialog box initialization
      Beep
      Debug.Print “DlgCount =”;DlgCount()3
   End Select
End Function
```

DlgEnable

Instruction/Function

Syntax

```
DlgEnable dlgitem|$[, enable]
-or-
DlgEnable(dlgitem|$)
```

Parameters

Name	Description
dlgitem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog items *field* name.
enable	It this numeric value is non-zero then enable *dlgItem|$*. Otherwise, disable it.

Description

Instruction: Enable or disable *DlgItem|$*.

Function: Return True if *DlgItem|$* is enabled.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Example

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button to exit."
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
      PushButton 110,90,60,20,"&Disable"
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub
Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print “Action =”;Action%
   Select Case Action%
   Case 1      ‘Dialog box initialization
      Beep
   Case 2      ‘Value changing Or button pressed
      Select Case DlgItem$
      Case “Disable”
         DlgText DlgItem$,"&Enable"
         DlgEnable Text,False
         DialogFunc% = True ‘Do not exit the dialog.
      Case “Enable”
```

```
            DlgText DlgItem$,"&Disable"
            DlgEnable Text,True
            DialogFunc% = True 'Do not exit the dialog.
        End Select
    End Select
End Function
```

DlgEnd

Instruction

Syntax

```
DlgEnd ReturnCode
```

Description

Set the return code for the Dialog Function and close the user dialog.

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameters

Parameter	Description
ReturnCode	Return this numeric value.

Example

```
Sub Main
    Begin Dialog UserDialog 210,120,.DialogFunc
            Text 10,10,190,15,"Please push the Close
button"
        OKButton 30,90,60,20
            CheckBox 120,90,60,20,"&Close",.CheckBox1
    End Dialog
    Dim dlg As UserDialog
    Debug.Print Dialog(dlg)
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
    Debug.Print "Action=";Action%
    Select Case Action%
    Case 1 ' Dialog box initialization
        Beep
    Case 2 ' Value changing or button pressed
        Select Case DlgItem$
        Case "CheckBox1"
            DlgEnd 1000
        End Select
```

```
    End Select
End Function
```

DlgFocus — Instruction/Function

Syntax

```
DlgFocus dlgitem|$
-or-
dlgfocus[$]()
```

Parameters

Name	Description
dlgitem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog items *field* name.

Description

Instruction: Move the focus to this *DlgItem|$*.

Function: Return the *field* name which has the focus as a string.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Example

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
      PushButton 110,90,60,20,"&Hello"
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub
Function DialogFunc%(DlgItem$,Action%,SuppValue%)
   Debug.Print "Action =";Action%
   Select Case Action%
   Case 1      'Dialog box initialization
      Beep
   Case 2      'Value changing Or button pressed
      If DlgItem$ = "Hello" Then
         MsgBox "Hello Button Pressed"
         DialogFunc% = True   'Do Not Exit the Dialog
      End If
```

```
   Case 4      'Focus changed
      Debug.Print "DlgFocus =""";DlgFocus();"""""
   End Select
End Function
```

Example Output

DlgListBoxArray — Instruction/Function

Syntax

DlgListBoxArray *dlgitem|$, strarray$()*
-or-
DlgListBoxArray(*dlgitem*|$[, *strarray$*()]**)**

Parameters

Name	Description
dlgitem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog items *field* name.
strarray$( )	Set the list entries of *DlgItem|$*. This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.

Description

Instruction: Set the list entries for *DlgItem|$*.

Function: Return the number entries in *DlgItem|$*s list.

This instruction/function must be called directly or indirectly from a *dialogfunc*. The *DlgItem|$* should refer to a **ComboBox**, **DropListBox** or **ListBox**.

Example

```
Dim lists$()
Sub Main
   ReDim lists$(0)
   lists$(0) = "List 0"
   Begin Dialog UserDialog 200,119,.DialogFunc
      Text 10,7,180,14,"Please push the OK button."
      ListBox 10,21,180,63,lists(),.list
      OKButton 30,91,40,21
      PushButton 110,91,60,21,"&Change"
   End Dialog
   Dim dlg As UserDialog
   dlg.list = 2
```

```
   Dialog dlg           'Show Dialog (Wait For ok)
   Debug.Print dlg.list
End Sub
Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Select Case Action%
   Case 2         'Value changing Or button pressed
      If DlgItem$ = "Change" Then
         Dim N As Integer
         N = UBound(lists$) + 1
         ReDim Preserve lists$(N)
         lists$(N) = "List " & N
         DlgListBoxArray "list",lists$()
         DialogFunc% = True   'Do Not Exit the Dialog
      End If
   End Select
End Function
```

Example Output

DlgName Function

Syntax **DlgName**[$]**(***dlgitem***)**

Parameters

Name	Description
dlgitem	This numeric value is the dialog item number. The first item is 0, second is 1, etc.

Description Return the *field* name of the *DlgItem* number. This instruction/function must be called directly or indirectly from a *dialogfunc*.

Example

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button.",.Text
      TextBox 10,40,180,15,.TextBox
      OKButton 30,90,60,20,.OKButton
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg
End Sub
Function DialogFunc%(DlgItem$, Action%, SuppValue%)
```

```
   Debug.Print "Action =";Action%
   Select Case Action%
   Case 1          'Dialog box initialization.
      Beep
      For I = 0 To DlgCount()-1
         Debug.Print I ;" = ";DlgName(I)
      Next I
   End Select
End Function
```

Example Output

```
Action = 1
 0 = Text
 1 = TextBox
 2 = OKButton
Action = 4
Action = 5
Action = 4
Action = 2
```

DlgNumber — Function

Syntax `DlgNumber(dlgitem$)`

Parameters

Name	Description
dlgitem$	This string value is the dialog items *field* name.

Description Return the number of the *DlgItem$*. The first item is 0, second is 1, etc. This instruction/function must be called directly or indirectly from a *dialogfunc*.

Example

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button."
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg
End Sub
Function DialogFunc%(DlgItem$, Action%, SuppValue%)
```

```
   Debug.Print "Action =";Action%
   Select Case Action%
   Case 1            'Dialog box initialization
      Beep
   Case 4            'Focus changed
      Debug.Print DlgItem$;" =";DlgNumber(DlgItem$)
   End Select
End Function
```

Example Output

```
Action = 1
Action = 4
Text = 1
Action = 5
Action = 4
OK = 2
Action = 2
```

DlgSetPicture — Instruction

Syntax: **DlgSetPicture** *DlgItem|$, FileName, Type*

Description Instruction: Set the file name for DlgItem|$.

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameters

Parameter	Description
DlgItem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's field name.
FileName	Set the file name of DlgItem|$ to this string value.
Type	This numeric value indicates the type of bitmap used. See below.

Type	Effect
0	FileName is the name of the bitmap file. If the file does not exist then "(missing picture)" is displayed.
3	The clipboard's bitmap is dispalyed. Not supported.
+16	Instead of displaying "(missing picture)" a run-time error occurs.

Example

```
Sub Main
    Begin Dialog UserDialog 200,120,.DialogFunc
        Picture 10,10,180,75,"",0,.Picture
        OKButton 30,90,60,20
        PushButton 110,90,60,20,"&View"
    End Dialog
    Dim dlg As UserDialog
    Debug.Print Dialog(dlg)
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
    Debug.Print "Action=";Action%
    Select Case Action%
    Case 1 ' Dialog box initialization
        Beep
    Case 2 ' Value changing or button pressed
        Select Case DlgItem$
        Case "View"
            FileName = GetFilePath("Bitmap","BMP")
            DlgSetPicture "Picture",FileName,0
            DialogFunc% = True 'do not exit the dialog
        End Select
    End Select
End Function
```

DlgText — Instruction/Function

Syntax

DlgText *dlgitem|$, text*
-or-
DlgText[$]**(***dlgitem|$***)**

Parameters

Name	Description
dlgitem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog items *field* name.
text	Set the text of *DlgItem|$* to this string value.

Description

Instruction: Set the text for *DlgItem|$*.

Function: Return the text from *DlgItem|$*.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Example

```
Sub Main
    Begin Dialog UserDialog 200,120,.DialogFunc
        Text 10,10,180,15,"Please push the OK button."
        TextBox 10,40,180,15,.Text
        OKButton 30,90,60,20
        PushButton 110,90,60,20,"&Now"
    End Dialog
    Dim dlg As UserDialog
    Debug.Print Dialog(dlg)
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
    Debug.Print “Action =”;Action%
    Select Case Action%
    Case 1
        Beep
    Case 2
        Select Case DlgItem$
        Case “Now”
            DlgText “Text”,CStr(Now)
            DialogFunc% = True ‘Do not exit the dialog
        End Select
    End Select
End Function
```

Example Output

```
Action = 1
Action = 4
Action = 5
Action = 4
Action = 2
-1
```

DlgType — Function

Syntax **DlgType**[$]**(***dlgitem*|*$***)**

Parameters

Name	Description
dlgitem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog items *field* name.

Description

Return a string value indicating the type of the *DlgItem|$*. One of: **CancelButton**, **CheckBox**, **ComboBox**, **DropListBox**, **GroupBox**, **ListBox**, **OKButton**, **OptionButton**, **OptionGroup**, **PushButton**, **Text**, **TextBox**.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Example

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button."
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg
End Sub
Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print Action=;Action%
   Select Case Action%
   Case 1  Dialog box initialization
      Beep
      For I = 0 To DlgCount()-1
         Debug.Print I;" “;DlgType(I)
      Next I
   End Select
End Function
```

Example Output

```
Action = 1
 0 Text
 1 TextBox
 2 OKButton
Action = 4
Action = 5
Action = 4
Action = 2
```

DlgValue

Instruction/Function

Syntax

```
DlgValue dlgitem|$, value
-or-
DlgValue(dlgitem|$)
```

Parameters

Name	Description
dlgitem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog items *field* name.
text	Set the text of *DlgItem|$* to this string value.

Description

Instruction: Set the numeric value *DlgItem|$*

Function: Return the numeric value for *DlgItem|$*.

This instruction/function must be called directly or indirectly from a *dialogfunc*. The *DlgItem|$* should refer to a **CheckBox**, **DropListBox**, **ListBox** or **OptionGroup**.

Example

```
Sub Main
   Begin Dialog UserDialog 150,147,.DialogFunc
      GroupBox 10,7,130,77,"Direction",.Field1
      PushButton 100,28,30,21,"&Up"
      PushButton 100,56,30,21,"&Dn"
      OptionGroup .Direction
         OptionButton 20,21,80,14,"&North",.North
         OptionButton 20,35,80,14,"&South",.South
         OptionButton 20,49,80,14,"&East",.East
         OptionButton 20,63,80,14,"&West",.West
      OKButton 10,91,130,21
      CancelButton 10,119,130,21
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg
   MsgBox "Direction = " & dlg.Direction
End Sub
Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Select Case Action%
   Case 1            'Dialog box initialization.
      Beep
   Case 2            'Value changing Or button pressed.
```

```
            Select Case DlgItem$
            Case “Up”
               DlgValue “Direction”,0
               DialogFunc% = True   ‘Do Not Exit the Dialog.
            Case “Dn”
               DlgValue “Direction”,1
               DialogFunc% = True   ‘Do Not Exit the dialog.
            End Select
         End Select
      End Function
```

DlgVisible

Instruction/Function

Syntax

```
DlgVisible dlgitem|$, visible
-or-
DlgVisible(dlgitem|$)
```

Parameters

Name	Description
dlgitem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog items *field* name.
enable	It this numeric value is non-zero then show *DlgItem|$*. Otherwise, hide it.

Description

Instruction: Show or hide *DlgItem|$*.

Function: Return *True* if *DlgItem|$* is visible.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Example

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
      PushButton 110,90,60,20,"&Hide"
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print “Action =”;Action%
   Select Case Action%
   Case 1            ‘Dialog box initialization
      Beep
   Case 2            ‘Value changing Or button pressed
      Select Case DlgItem$
      Case “Hide”
         DlgText DlgItem$,"&Show"
         DlgVisible “Text”,False
         DialogFunc% = True   ‘Do Not Exit the Dialog
```

```
        Case "Show"
           DlgText DlgItem$,"&Hide"
           DlgVisible "Text",True
           DialogFunc% = True   'Do Not Exit the Dialog
        End Select
     End Select
End Function
```

Do Statement

Syntax

```
Do
  statements
Loop
-or-
Do {Until|While} condexpr
  statements
Loop
-or-
Do
  statements
Loop {Until|While} condexpr
```

Description

Form 1: Do *statements* forever. The loop can be exited by using **Exit** or **Goto**.

Form 2: Check for loop termination before executing the loop the first time.

Form 3: Execute the loop once and then check for loop termination.

Loop Termination:

Until *condexpr*: Do *statements* until *condexpr* is **True**.

While *condexpr*: Do *statements* while *condexpr* is **True**.

See Also

For, For Each, Exit Do, **While.**

Example

```
Sub Main
    I = 2
```

```
    Do
        I = I*2
    Loop Until I > 10
    Debug.Print I
End Sub
```

Example Output 16

DoEvents — Instruction

Syntax **DoEvents**

Description This instruction allows other applications to process events.

Example

```
Sub Main
   DoEvents        'let other apps work
End Sub
```

DropListBox Dialog Item — Definition

Syntax **DropListBox** *x, y, dx, dy, strarray$( ), .field*

Parameters

Name	Description
x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.
y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
strarray$( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
field	The value of the drop-down list box is accessed via this field. It is the index of the *StrArray$( )* var.

Description Define a drop-down listbox item.

See Also `Begin Dialog, Dim As UserDialog.`

Example

```
Sub Main
   Dim lists$(3)
   lists$(0) = “List 0"
   lists$(1) = “List 1"
   lists$(2) = “List 2"
   lists$(3) = “List 3"
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button."
      DropListBox 10,25,180,60,lists$(),  .list
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   dlg.list = 2
   Dialog dlg        ‘show Dialog (Wait For OK)
   Debug.Print dlg.list
End Sub
```

End — Instruction

Syntax **End**

Description The end instruction causes the macro to terminate immediately. If the macro was run by another macro using the **MacroRun** instruction then that macro continues on the instruction following the **MacroRun.**

Example

```
Sub DoSub
   L$ = UCase$(“InputBox$ (Enter End:)”)
   If L$ = “END” Then End
   Debug.Print “End was Not entered.”
End Sub
Sub Main
   Debug.Print “Before DoSub”
   DoSub
   Debug.Print “After DoSub”
End Sub
```

Example Output

```
Before DoSub
```

```
End was Not entered.
After DoSub
```

Enum

Definition

Syntax

```
[ | Private | Public ] Enum name
    elem [ = value]
   [...]
End Enum
```

Description

Define a new userenum. Each elem defines an element of the enum. If value is given then that is the element's value. The value can be any constant integer expression. If value is omitted then the element's value is one more than the previous element's value. If there is no previous element then zero is used.

Enum defaults to Public if neither Private or Public is specified.

Example

```
Enum Days
    Monday
    Tuesday
    Wednesday
    Thursday
    Friday
    Saturday
    Sunday
End Enum

Sub Main
    Dim D As Days
    For D = Monday To Friday
        Debug.Print D ‘ 0 through 4
    Next D
End Sub
```

Environ

Instruction/Function

Syntax

Environ[*$*](*Index*)

-or-

Environ[*$*](*Name*)

Description

Return an environment string.

Parameters

Parameter	Description
Index	Return this environment string's value. If there is no environment string at this index a null string is returned. Indexes start at one.
Name	Return this environment string's value. If the environment string can't be found a null string is returned.

Example

```
Sub Main
    Debug.Print Environ("Path")
End Sub
```

Eof

Function

Syntax

Eof(*streamnum*)

Parameters

Name	Description
streamnum	Streams 1, 2, 3 and 4 are available in each macro.

Description

Return *True* if *Streamnum* is at the end of the file.

Example

```
Sub Main
   Open XXX For Input As #1
   While Not Eof(1)
      Line Input #1,L$
      Debug.Print L$
   Wend
   Close #1
End Sub
```

Erase — Instruction

Syntax `Erase array[, ...]`

Description Reset *array* to zero. (Dynamic arrays are reset to undimensioned arrays.) String arrays values are set to a null string. *Array* must be declared as an array using **Dim, Private, Public** or **Static.**

Example

```
Sub Main
   Dim X%(2)
   X%(1) = 1
   Erase X%
   Debug.Print X%(1) '" 0"
End Sub
```

Example Output

```
 0
```

Err — Variable

Syntax `Err = errorcode`

Description Set it to zero to clear the last error condition. Use **Error** to trigger an error event. Err in an expression returns the current error code.

Example

```
Sub Main
   On Error GoTo Problem
   Error 1     'simulate Error #1
   Exit Sub
   Problem: 'Error handler
   Debug.Print "Error Number =";Err
   Debug.Print "Error String = ";Error$
   Resume Next
End Sub
```

Example Output

```
Error Number = 1
Error String = Application specific error #1.
```

Error

Instruction/Function

Syntax

```
Error errorcode
-or-
Error[$]([errorcode])
```

Parameters

Name	Description
errorcode	This is the error number.

Description

Instruction: Signal error *ErrorCode*. This triggers error handling just like a real error. The current procedures error handler is activated, unless it is already active or there isnt one. In that case the calling procedures error handler is tried. If no available error handler is found in any of the calling procedures of the current macro, the macro is halted.

Function: The Error( ) function returns the error text string.

Example

```
Sub Main
   On Error GoTo Problem
   Error 1     'simulate Error #1
   Exit Sub
   Problem: 'Error handler
   Debug.Print "Error Number =";Err
   Debug.Print "Error String = ";Error$
   Resume Next
End Sub
```

Example Output

```
Error Number = 1
Error String = Application specific error #1.
```

Exit

Instruction

Syntax

```
Exit {All|Do|For|Function|Property|Sub|While}
```

Parameters

Exit	Description
All	Exit all macros.
Do	Exit the **Do** loop.
For	Exit the **For** of **For Each** loop.

Function	Exit the Function block. Note: This instruction resets **Err** to zero and **Error$** to null.
Property	Exit the **Property** block. Note: This instruction resets **Err** to zero and **Error$** to null .
Sub	Exit the **Sub** block. Note: This instruction resets **Err** to zero and **Error$** to null.
While	Exit the **While** loop.

Description

The exit instruction causes the macro to continue without doing some or all of the remaining instructions.

Example

```
Sub DoSub(L$)
   Do
      If L$ = “DO” Then Exit Do
   I = I+1
   Loop While I < 10
   If I = 0 Then Debug.Print “Do was entered”
   For I = 1 To 10
      If L$ = “FOR” Then Exit For
   Next I
      If I = 1 Then Debug.Print “For was entered”
   I = 10
   While I > 0
      If L$ = “WHILE”   Then Exit While
      I = I-1
   Wend
   If I = 10 Then Debug.Print “While was entered”
   If L$ = “SUB” Then Exit Sub
   Debug.Print “Sub was Not entered.”
   If L$ = “ALL” Then Exit All
   Debug.Print “All was Not entered.”
End Sub
Sub Main
   L$ = InputBox$(“Enter Do, For, While,Sub Or All:”)
   Debug.Print “Before DoSub”
   DoSub UCase$(L$)
   Debug.Print “After DoSub”
End Sub
```

Example Output

```
Before DoSub
Do was entered
Sub was Not entered.
```

```
All was Not entered.
After DoSub
```

Exp — Function

Syntax **Exp(***num***)**

Parameters

Name	Description
num	Return e raised to the power of this number value. The value e is approximately 2.71 8282.

Description The **Exp** function computes the exponential of the variable *num*.

Example

```
Sub Main
   Debug.Print Exp(1)
End Sub
```

Example Output

```
2.71828182845905
```

Exp10 — Function

Syntax **Exp10(***num***)**

Parameters

Name	Description
num	Return 10 raised to the power of this number value.

Description The **Exp10** function computes the base-10 exponential of the variable *num*.

Example

```
Sub Main
   Debug.Print Exp10(1)
End Sub
```

Example Output

```
10
```

FileAttr — Function

Syntax `FileAttr(StreamNum, ReturnValue)`

Description Return StreamNum's open mode or file handle.

ParameterDescription

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

ReturnValue 1 - return the mode used to open the file: 1=Input, 2=Output, 4=Random, 8=Append, 32=Binary

2 - return the file handle

See Also Open.

Example

```
Sub Main
    Open "XXX" For Output As #1
    Debug.Print FileAttr(1,1) ' 2
    Close #1
End Sub
```

FileCopy — Instruction

Syntax `FileCopy FromName$, ToName$`

Description Copy a file.

Parameters

Parameter	Description
FromName$	This string value is the path and name of the source file. A path relative to the current directory can be used.
ToName$	This string value is the path and name of the destination file. A path relative to the current directory can be used.

Example

```
Sub Main
    FileCopy "C:\AUTOEXEC.BAT","C:\AUTOEXEC.BAK"
End Sub
```

FileDateTime

Function

Syntax `FileDateTime(name$)`

Parameters

Name	Description
name$	This string value is the path and name of the file. A path relative to the current directory can be used.

Description Return the date and time file *Name$* was last changed as a *date* value. If the file does not exist then a run-time error occurs.

Example

```
Sub Main
   F$ = Dir$("*.*")
   While F$ <> ""
      Debug.Print F$;" ";"";FileDateTime(F$)
      F$ = Dir$()
   Wend
End Sub
```

Example Output

```
SNR.APB 12/22/95 4:21:06 PM
FRQ-RESP.AT1 12/22/95 4:21:06 PM
```

FileLen — Function

Syntax `FileLen(name$)`

Parameters

Name	Description
name$	This string value is the path and name of the file. A path relative to the current directory can be used.

Description Return the length of file *Name$*. If the file does not exist then a run-time error occurs.

Example

```
Sub Main
   F$ = Dir$("*.*")
   While F$ <> ""
      Debug.Print F$;" ";"";FileLen(F$)
      F$ = Dir$()
   Wend
End Sub
```

Example Output

```
SNR.APB  311
FRQ-RESP.AT1  31744
```

Fix — Function

Syntax `Fix(num)`

Parameters

Name	Description
num	Return the integer portion of this number value. The number is truncated. Positive numbers return the next lower integer. Negative numbers return the next higher integer.

Description Return the integer value.

Example

```
Sub Main
   Debug.Print Fix(9.9)
   Debug.Print Fix(0)
   Debug.Print Fix(-9.9)
End Sub
```

Example Output

```
 9
 0
```

```
-9
```

For Statement

Syntax

```
For num = first To last [step Inc]
  statements
Next [num]
```

Parameters

Name	Description
num	This is the iteration variable.
first	Set *num* to this value initially.
last	Continue looping while *num* is in the range. See *Step* below.
step	If this number value is greater than zero then the for loop continues as long as *num* is less than or equal to *Last*. If this number value is less than zero then the for loop continues as long as *num* is greater than or equal to *Last*. If this is omitted then one is used.

Description

Execute *statements* while *num* is in the range *First* to *Last*.

See Also

`Do, For Each, Exit For, While.`

Example

```
Sub Main
   For I = 0 To 300 Step 100
      Debug.Print I;I+I;I*I
   Next I
End Sub
```

Example Output

```
 0 0 0
 100 200 10000
 200 400 40000
 300 600 90000
```

For Each Statement

Syntax

```
For Each var In items
  statements
Next [var]
```

Parameters

Name	Description
var	This is the iteration variable.
items	This is the collection of items to be done.

Description Execute *statements* for each item in *Items*.

See Also `Do, For, Exit For, While.`

Example

```
Sub Main
   Dim Document As Object
   For Each Document In MicroSoft.Word.Documents
      Debug.Print Document.Title
   Next Document
End Sub
```

Format$ Function

Syntax **`Format`**`[$]`**`(`***`expr`***`[, `***`form$`***`]`**`)`**

Description Return the formatted string representation of *expr*.

Parameters

Name	Description
expr	Return the formatted string representation of this number value.
form	Format *expr* using to this string value. If this is omitted then return the *expr* as a string. See below: Predefined Date Format, Predefined Number Format, User defined Date Format, User defined Number Format, User defined Text Format.

Format Predefined Date

Description The following predefined date formats may be used with the **Format** function. Predefined formats may not be combined with user defined formats or other predefined formats.

Form	Description
General Date	Same as **user defined date format** "c"
Long Date	Same as **user defined date format** "dddddd"
Medium Date	Not supported at this time.

Short Date	Same as **user defined date format** "ddddd"
Long Time	Same as **user defined date format** "ttttt"
Medium Time	Same as **user defined date format** "hh:mm AMPM "
Short Time	Same as **user defined date format** "hh:mm"

Format Predefined Number

Description

The following predefined number formats may be used with the **Format** function. Predefined formats may not be combined with user defined formats or other predefined formats.

Form	Description
General number	Return number as is.
Currency	Same as **user defined number format** " $#,##0.00;($#,##0.00)" **Not locale dependent at this time.**
Fixed	Same as **user defined number format** "0.00".
Standard	Same as **user defined number format** "#,##0.00".
Percent	Same as **user defined number format** "0.00%".
Scientific	Same as **user defined number format** "0.00E+00".
Yes/No	Return No if zero, else return "Yes".
True/False	Return True if zero, else return "False".
On/Off	Return On if zero, else return "Off".

Example

```
Sub Main
   Debug.Print Format$(2.145,"Standard")
End Sub
```

Example Output

```
2.15
```

Format User Defined Date

Description

The following date formats may be used with the **Format** function. Date formats may be combined to create the user defined date format. User defined date formats may not be combined with other user defined formats or predefined formats.

Form	Description
:	insert localized time separator
/	insert localized date separator

c	insert ddddd ttttt, insert date only if t=0, insert time only if d=0
d	insert day number without leading zero
dd	insert day number with leading zero
ddd	insert abbreviated day name
dddd	insert full day name
ddddd	insert date according to Short Date format
dddddd	insert date according to Long Date format
w	insert day of week number
ww	insert week of year number
m	insert month number without leading zero insert minute number without leading zero (if follows h or hh)
mm	insert month number with leading zero insert minute number with leading zero (if follows h or hh)
mmm	insert abbreviated month name
mmmm	insert full month name
q	insert quarter number
y	insert day of year number
yy	insert year number (two digits)
yyyy	insert year number (four digits, no leading zeros)
h	insert hour number without leading zero
hh	insert hour number with leading zero
n	insert minute number without leading zero
nn	insert minute number with leading zero
s	insert second number without leading zero
ss	insert second number with leading zero
ttttt	insert time according to time format
AM/PM	use 12 hour clock and insert AM (hours 0 to 11) and PM (12 to 23)
am/pm	use 12 hour clock and insert am (hours 0 to 11) and pm (12 to 23)
A/P	use 12 hour clock and insert A (hours 0 to 11) and P (12 to 23)
a/p	use 12 hour clock and insert a (hours 0 to 11) and p (12 to 23)
AMPM	use 12 hour clock and insert localized AM/PM strings
\c	insert character c
"text"	insert literal text

Example

Format User Defined Number

Description

The following number formats may be used with the **Format** function. Number formats may be combined to create the user defined number format. User defined number formats may not be combined with other user defined formats or predefined formats.
User defined number formats can contain up to four sections separated by ;:

form - format for non-negative expr, -format for negative expr, empty and null expr return

form;negform - negform: format for negative expr

form;negform;zeroform - zeroform: format for zero expr

form;negform;zeroform;nullform - nullform: format for empty or null expr

Form	Description
#	digit, don't include leading/trailing zero digits (all the digits left of decimal point are returned) eg. Format(19,"###") returns "19" eg. Format(19,"#") returns "19"
0	digit, include leading/trailing zero digits eg. Format(19,"000") returns "019" eg. Format(19,"0") returns "19"
.	decimal, insert localized decimal point eg. Format(19.9,"###.00") returns "19.90" eg. Format(19.9,"###.##") returns "19.9"
,	thousands, insert localized thousand separator every 3 digits xxx, or xxx,. mean divide expr by 1000 prior to formatting two adjacent commas ",," means divide expr by 1000 again eg. Format(1900000,"0,,") returns "2" eg. Format(1900000,"0,,.0") returns "1.9"
%	percent, insert %, multiply expr by 100 prior to formatting
:	insert localized time separator
/	insert localized date separator

E+ e+ E- e-	use exponential notation, insert E (or e) and the signed exponent eg. Format(1000,"0.00E+00") returns “1.00E+03" eg. Format(.001,”0.00E+00") returns “1.00E-03"
+ $ ( )space	insert literal char eg. Format(10,"$#") returns “$10"
\c	insert character c eg. Format(19,"\####\#") returns “#19#”
“text”	insert literal text eg. Format(19,""""##""###""##""") returns “##19##”

Example

```
Sub Main
   Debug.Print Format$(2.145,"#.00")
End Sub
```

Example Output

```
2.15
```

Format User Defined Text

Description

The following text formats may be used with the **Format** function. Text formats may be combined to create the user defined text format. User defined text formats may not be combined with other user defined formats or predefined formats.
User defined text formats can contain one or two sections separated by ‘;’:

form - format for all strings
form;nullform - nullform: format for null strings

Form	Description
@	char placeholder, insert char or space
&	char placeholder, insert char or nothing
<	all chars lowercase
>	all chars uppercase
!	fill placeholder from left-to-right (default is right-to-left)
\c	insert character c
“text”	insert literal text

Example

```
Sub Main
   Debug.Print Format(“123",”ab@c")
   Debug.Print Format(“123",”!ab@c")
```

```
End Sub
```

Example Output

```
12ab3c
ab1c23
```

FreeFile — Instruction

Syntax

```
FreeFile[( )]
```

Description

Return the next unused stream number. Streams 1, 2, 3 and 4 are available in each macro.

Example

```
Sub Main
   Debug.Print FreeFile  '1
   Open XXX For Input As #1
   Debug.Print FreeFile  '2
   Close #1
   Debug.Print FreeFile  '1
End Sub
```

Example Output

Function — Definition

Syntax

```
[Private|Public] Function name[type][([param[, _
…]])] [As type]
  statements
End Function
```

Description

User defined function. The function defines a set of *statements* to be executed when it is called. The values of the calling *arglist are assigned to the parameters in the params. Assigning to* name[*type*] sets the value of the function result.

Public is assumed if neither *Private* or *Public* is specified.

See Also

`Declare, Property, Sub.`

Example

```
Function Power(X,Y)
   P = 1
```

```
   For I = 1 To Y
      P = P*X
   Next I
   Power = P
End Function
Sub Main
   Debug.Print Power(2,8)
End Sub
```

Example Output

```
 256
```

Get — Instruction

Syntax

```
Get StreamNum, [RecordNum], var
```

Parameters

Name	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1. If this is omitted then the current position (or record number) is used.
var	This variable value is read from the file. For a fixed length variable (like Long) the number of bytes required to restore the variable are read. For a Variant variable two bytes are read which describe its type and then the variable value is read accordingly. For a usertype variable each field is read in sequence. For an array variable each element is read in sequence. For a dynamic array variable the number of dimensions and range of each dimension is read prior to reading the array values. All binary data values are read from the file in little-endian format.

Note: When reading a string (or a dynamic array) from a Binary mode file the length (or array dimension) information is not read. The current string length determines how much string data is read. The current array dimension determines how may array elements are read.

Description

Get a variable's value from StreamNum.

See Also Open, Put.

Example

```
Sub Main
    Dim V As Variant
    Open “SAVE_V.DAT” For Binary Access Read As #1
    Get #1, , V
    Close #1
End Sub
```

GetAllSettings — Function

Syntax **GetAllSettings(***AppName$, Section$, Key$***)**

Parameters

Name	Description
AppName$	This string value is the name of the project which has this Section and Key.
Section$	This string value is the name of the section of the project settings.

Description Get all of Section’s settings in project AppName. Settings are returned in a Variant. Empty is returned if there are no keys in the section. Otherwise, the Variant contains a two dimension array: (I,0) is the key and (I,1) is the setting. Win16 and Win32s store settings in a .ini file named AppName. Win32 stores settings in the registration database.

Example

```
Sub Main
    SaveSetting “MyApp”,"Font","Size",10
    SaveSetting “MyApp”,"Font","Name","Courier"
    Settings = GetAllSettings(“MyApp”,"Font")
    For I = LBound(Settings) To UBound(Settings)
        Debug.Print Settings(I,0); “=”; Settings(I,1)
    Next I
    DeleteSetting “MyApp”,"Font"
End Sub
```

GetAttr — Function

Syntax **GetAttr(***Name$***)**

Parameters

Name	Description
Name$	This string value is the path and name of the file. A path relative to the current directory can be used.

Description

Return the *attributes* for file *Name$*. If the file does not exist then a run-time error occurs.

Example

```
Sub Main
   F$ = Dir$("*.*")
   While F$ <> ""
      Debug.Print F$;"";GetAttr(F$)
      F$ = Dir$()
   Wend
End Sub
```

Example Output

```
SNR.APB 32
FRQ-RESP.AT1 32
```

GetFilePath$ Function

Syntax

GetFilePath[$]([*defname$*], [*defext$*], [*defdir$*], _
[*title$*], [*option*])

Parameters

Name	Description
defname$	Set the initial File Name to this string value. If this is omitted then *.*DefExt$* is used.
defext$	Initially show files whose extension matches this string value. (Multiple extensions can be specified by using ";" as the separator.) If this is omitted then * is used.
defdir$	This string value is the initial directory. If this is omitted then the current directory is used.
title$	This string value is the title of the dialog. If this is omitted then Open" is used.
option	This numeric value determines the file selection options. If this is omitted then zero is used. See table below.

Option	Effect
0	Only allow the user to select a file that exists.

1	Confirm creation when the user selects a file that does not exist.
2	Allow the user to select any file whether it exists or not.
3	Confirm overwrite when the user selects a file that exists.

Description Put up a dialog box and get a file path from the user. The returned string is a complete path and file name. If the cancel button is pressed then a null string is returned.

Example

```
Sub Main
   Debug.Print GetFilePath$("*.*")
End Sub
```

Example Output

```
C:\APWIN\Samples\S1\Snr.apb
```

GetObject — Function

Syntax **GetObject(***file$*[, *class$*]**)**

Parameters

Name	Description
filename$	This is the file where the object resides. If this is omitted then the currently active object for *Class$* is returned.
class$	This string value is the applications registered class name. If this application is not cu rrently active it will be started. If this is omitted then the application associated with the files extension will be started.

Description Get an existing object of type *Class$* from *File$*. Use **Set** to assign the returned object to an object variable.

Example

```
Sub Main
   Dim App As Object
   Set App = GetObject(,"??????.Application")
   App.Move 20,30  move icon to 20,30
   Set App = Nothing
   App.Quit  'run-time error (no object)
End Sub
```

GetSetting — Function

Syntax `GetSetting[$](AppName$, Section$, Key$)`

Description Get the setting for Key in Section in project AppName. Win16 and Win32s store settings in a .ini file named AppName. Win32 stores settings in the registration database.

Parameter	Description
AppName$	This string value is the name of the project which has this Section and Key.
Section$	This string value is the name of the section of the project settings.
Key$	This string value is the name of the key in the section of the project settings.

Example

```
Sub Main
    SaveSetting “MyApp”,"Font","Size",10
    Debug.Print GetSetting(“MyApp”,"Font","Size") ‘ 10
End Sub
```

Goto — Instruction

Syntax `GoTo label`

Description Go to the *label* and continue execution from there. Only *labels* in the current user subroutine. Function or property are accessible.

Example

```
Sub Main
   X = 2
Label:
   X = X*X
   If X <= 100 Then GoTo Label
   Debug.Print X
End Sub
```

Example Output

```
 256
```

GroupBox Dialog Item

Definition

Syntax **GroupBox** *x, y, dx, dy, title$*[, *.field*]

Parameters

Name	Description
x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.
y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
title$	This string value is the title of the group box.
field	This identifier is the name of the *field*. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used.

Description Define a groupbox item.

See Also **Begin Dialog, Dim** As *UserDialog*.

Example

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button."
      GroupBox 10,25,180,60,"Group box"
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg        'Show Dialog (Wait For OK)
End Sub
```

Hex$

Function

Syntax **Hex**[$]**(***num***)**

Parameters

Name	Description

	num	Return a hex encoded string for this number value.

Description Return a hex string.

See Also `Oct$( ), Str$( ), Val( ).`

Example

```
Sub Main
   Debug.Print Hex$(15)
End Sub
```

Example Output

```
F
```

Hour — Function

Syntax **Hour(***dateexpr***)**

Parameters

Name	Description
dateexpr	Return the hour of the day for this date value.

Description Return the hour of the day (0 to 23).

See Also `Minute( ), Second( ), Time( ).`

Example

```
Sub Main
   Debug.Print Hour(#12:00:01 AM#)
End Sub
```

Example Output

```
 0
```

If — Statement

Syntax

If *condexpr* **Then** [*instruction*] [Else *instruction*]

-or-

```
If condexpr Then
  statements
[ElseIf condexpr Then
  statements...]
[Else
```

```
   statements]
End If
```

Description

Form 1: Single line if statement. Execute the *instruction* following the Then if *condexpr* is **True**. Otherwise, execute the *instruction* following the Else. The Else portion is optional.

Form 2: The multiple line if is useful for complex ifs. Each if *condexpr* is checked in turn. The first **True** one causes the following *statements* to be executed. If all are **False** then the Elses *statements* are executed. The ElseIf and Else portions are optional.

See Also

`Select Case, Choose( ), IIf( ).`

Example

```
Sub Main
   S = InputBox("Enter hello, goodbye, dinner Or sleep:")
   S = UCase(S)
   If S = "HELLO" Then Debug.Print "Come In"
   If S = "GOODBYE" Then Debug.Print "See you later"
   If S = "DINNER" Then
      Debug.Print "Please come In."
      Debug.Print "Dinner will be ready soon."
   ElseIf S = "SLEEP" Then
      Debug.Print "Sorry."
      Debug.Print "We are full For the night"
   End If
End Sub
```

IIf — Function

Syntax

IIf(*condexpr*, *truepart*, falsepart**)**

Parameters

Name	Description
condexpr	If this value is true then return *TruePart*. Otherwise, return *FalsePart.*
truepart	Return this value if *condexpr* is *True*.
falsepart	Return this value if *condexpr* is *False.*

Description

Return the value of the indicated by *condexpr*. Both *TruePart* and *FalsePart* are evaluated.

See Also `If, Select Case, Choose( ).`

Example

```
Sub Main
   Debug.Print IIf(1 > 0,"True","False")
End Sub
```

Example Output

```
True
```

Input — Instruction

Syntax **Input** [#]*streamnum*, *var*[, ...]

Description Get input from *Streamnum* and assign it to *vars*. Input values are comma delimited. Leading and trailing spaces are ignored. If the first char (following the leading spaces) is a quote (“) then the string is terminated by an ending quote. Special values #NULL#, #FALSE#, #TRUE#, #date# and #Error number# are converted to their appropriate value and data type.

See Also `Line Input, Print, Write.`

Example

```
Sub Main
   Open XXX For Input As #1
   Input #1,A,B,C$
   Debug.Print A;B;C$
   Close #1
End Sub
```

Input$ — Function

Syntax **Input**[$](*n*, *streamnum*)

Parameters

Name	Description
n	Read *n* chars. If fewer than *n* chars are left before the end of file then a run-time error occurs.
streamnum	Streams 1, 2, 3 and 4 are available in each macro.

Description Return *N* chars from *Streamnum*.

Example

```
Sub Main
   Open XXX For Input As #1
   L = Lof(1)
   T$ = Input$(L,1)
   Close #1
   Debug.Print T$;
End Sub
```

InputBox$ Function

Syntax

InputBox[$](*Prompt$*[, *title$*]
[, *default$*][, *xpos*, *ypos*])

Parameters

Name	Description
prompt$	Use this string value as the prompt in the input box.
title$	Use this string value as the title of the input box. If this is omitted then the input box does not have a title.
default$	Use this string value as the initial value in the input box. If this is omitted then the initial value is blank.
xpos	When the dialog is put up the left edge will be at this screen position. If this is omitted then the dialog will be centered.
ypos	When the dialog is put up the top edge will be at this screen position. If this is omitted then the dialog will be centered.

Description

Display an input box where the user can enter a line of text. Pressing the OK button returns the string entered. Pressing the Cancel button returns a null string.

Example

```
Sub Main
   L$ = InputBox$(“Enter some Text:”,"Input Box
⇨Example","Example text")
   Debug.Print L$
End Sub
```

Example Output

```
Example text
```

InStr Function

Syntax `InStr([index, ]String1$, String2$)`

Parameters

Name	Description
index	Start searching for *S2$* at this offset in *S1$*. If this is omitted then start searching from the beginning of *S1$*.
string1$	Search for *S2$* in this string value.
string2$	Search *S1$* for this string value.

Description Return the index where *S2$* first matches *S1$*. If no match is found return 0.

See Also Left$( ), Len( ), Mid$( ), Right$( ).

Example

```
Sub Main
   Debug.Print InStr("Hello","l")
End Sub
```

Example Output

```
3
```

InStrRev Function

Syntax `InStrRev(S1$, S2$[, Index])`

Description Return the index where S2$ last matches S1$. If no match is found return 0.

Parameters

Name	Description
S1$	Search for S2$ in this string value. If this value is Null then Null is returned.
S2$	Search S1$ for this string value. If this value is Null then Null is returned.
Index	Start searching for S2$ ending at this index in S1$. If this is omitted then start searching from the end of S1$.

See Also Left$( ), Len( ), Mid$( ), Replace$( ), Right$( ).

Example

```
Sub Main
    Debug.Print InStrRev(“Hello”,"l") ‘ 4
End Sub
```

Int — Function

Syntax `Int(num)`

Parameters

Name	Description
num	Return the largest integer which is less than or equal to this number value.

Description Return the integer value.

Example

```
Sub Main
   Debug.Print Int(9.9)
   Debug.Print Int(0)
   Debug.Print Int(-9.9)
End Sub
```

Example Output

```
 9
 0
-10
```

Is — Operator

Syntax *expr* ***Is*** *expr*

Description Return the *True* if both *exprs* refer to the same object.

See Also Objects.

Example

```
Sub Main
   Dim X As Object
   Dim Y As Object
   Debug.Print X Is Y
End Sub
```

Example Output

```
True
```

IsArray — Function

Syntax `IsArray(var)`

Parameters

Name	Description
var	A array variable or a variant var can contain multiple values.

Description Return the *True* if *var* is an array of values.

See Also `TypeName, VarType.`

Example

```
Sub Main
   Dim X As Variant, Y(2) As Integer
   Debug.Print IsArray(X)
   X = Array(1,4,9)
   Debug.Print IsArray(X)
   X = Y
   Debug.Print IsArray(X)
End Sub
```

Example Output

```
False
True
True
```

IsDate — Function

Syntax `IsDate(expr)`

Parameters

Name	Description
expr	A variant expression to test for a valid date.

Description Return the *True* if *expr* is a valid date.

See Also `TypeName, VarType.`

Example

```
Sub Main
   Dim X As Variant
   X = 1
   Debug.Print IsDate(X)
   X = Now
   Debug.Print IsDate(X)
```

```
End Sub
```

Example Output

```
False
True
```

IsEmpty — Function

Syntax `IsEmpty(variantvar)`

Parameters

Name	Description
variantvar	A variant var is *Empty* if it has never been assigned a value.

Description Return the *True* if *variantvar* is *Empty*.

See Also `TypeName, VarType.`

Example

```
Sub Main
   Dim X As Variant
   Debug.Print IsEmpty(X)
   X = 0
   Debug.Print IsEmpty(X)
   X = Empty
   Debug.Print IsEmpty(X)
End Sub
```

Example Output

```
True
False
True
```

IsError — Function

Syntax `IsError(expr)`

Parameters

Name	Description
expr	A variant expression to test for an error code value.

Description Return the *True* if *expr* is an error code.

See Also `TypeName, VarType.`

Example

```
Sub Main
   Dim X As Variant
   Debug.Print IsError(X)
   X = CVErr(1)
   Debug.Print IsError(X)
End Sub
```

Example Output

```
False
True
```

IsMissing — Function

Syntax **IsMissing(***variantvar***)**

Parameters

Name	Description
variantvar	Return True if this parameters argument expression was not specified in the **Sub, Function** or **Property** call.

Description

Return the *True* if Optional parameter *variantvar* did not get a value. An Optional or ParamArray parameter may be omitted in the **Sub, Function** or **Property** call.

Example

```
Sub Main
    Opt              'IsMissing(A)=True
    Opt "Hi"         'IsMissing(A)=False
    Many             'No args
    Many 1,"Hello"   'A(0)=1 A(1)=Hello
End Sub
Sub Opt(Optional A)
    Debug.Print "IsMissing(A) = ";IsMissing(A)
End Sub
Sub Many(ParamArray A())
    If LBound(A) > UBound(A) Then
        Debug.Print "No args"
    Else
        For I = LBound(A) To UBound(A)
            Debug.Print "A(" & I & ") = " & A(I) & " "
        Next I
        Debug.Print
    End If
```

```
End Sub
```

Example Output

```
IsMissing(A) = True
IsMissing(A) = False
No args
A(0) = 1
A(1) = Hello
```

IsNull — Function

Syntax `IsNull(expr)`

Parameters

Name	Description
expr	A variant expression to test for *Null*.

Description Return the *True* if *expr* is *Null*.

See Also `TypeName, VarType.`

Example

```
Sub Main
    Dim X As Variant
    Debug.Print IsNull(X) '(IsEmpty, but not IsNull)
    X = 1
    Debug.Print IsNull(X)
    X = "1"
    Debug.Print IsNull(X)
    X = Null
    Debug.Print IsNull(X)
    X = X*2
    Debug.Print IsNull(X)
End Sub
```

Example Output

```
False
False
False
True
True
```

IsNumeric — Function

Syntax `IsNumeric(expr)`

Parameters

Name	Description
expr	A variant expression is a numeric value if it is *numeric* or string value that represents a number.

Description Return the *True* if *expr* is a numeric value.

See Also `TypeName, VarType.`

Example

```
Sub Main
   Dim X As Variant
   X = 1
   Debug.Print IsNumeric(X)
   X = "1"
   Debug.Print IsNumeric(X)
   X = "A"
   Debug.Print IsNumeric(X)
End Sub
```

Example Output

```
True
True
False
```

IsObject — Function

Syntax `IsObject(var)`

Parameters

Name	Description
var	A var contains an object reference if it is *objexpr* reference.

Description Return the *True* if *var* contains an object reference.

See Also `TypeName, VarType.`

Example

```
Sub Main
   Dim X As Variant
   X = 1
```

```
    Debug.Print IsObject(X)
    X = 1"
    Debug.Print IsObject(X)
    Set X = Nothing
    Debug.Print IsObject(X)
End Sub
```

Example Output

```
False
False
True
```

Kill — Instruction

Syntax

```
Kill Name$
```

Parameters

Name	Description
name$	This string value is the path and name of the file. A path relative to the current directory can be used.

Description

Delete the file named by *name$*.

Example

```
Sub Main
   Kill "FILENAME.EXT"
End Sub
```

LBound — Function

Syntax

```
LBound(var[, dimension])
```

Parameters

Name	Description
var	Return the lowest index for this array variable.
dimension	Return the lowest index for this dimension of *var*. If this is omitted then return the lowest index for the first dimension.

Description

Return the lowest index.

See Also

UBound( ).

Example

```
Sub Main
   Dim A(-1 To 3,2 To 6)
   Debug.Print LBound(A)
   Debug.Print LBound(A,1)
   Debug.Print LBound(A,2)
End Sub
```

Example Output

```
-1
-1
 2
```

LCase$ Function

Syntax **LCase**[$]**(***string$***)**

Parameters

Name	Description
string$	Return the string value of this after all chars have been converted to lowercase.

Description Return a string from *string$* where all the uppercase letters have been lowercased.

See Also UCase$( ).

Example

```
Sub Main
   Debug.Print LCase$("Hello")
End Sub
```

Example Output

```
hello
```

Left$ Function

Syntax **Left**[$]*(*string$*,* *len*)*

Parameters

Name	Description
string$	Return the left portion of this string value.
len	Return this many chars. If *string$* is shorter than that then just return *string$*.

Description Return a string from *S$* with only the *Len* chars.

See Also `InStr( ), Len( ), Mid$( ), Right$( ).`

Example

```
Sub Main
   Debug.Print Left$("Hello",2)
End Sub
```

Example Output

```
He
```

Len — Function

Syntax **`Len(`***`string$`***`)`**

Parameters

Name	Description
string$	Return the number of chars in this string value.

Description Return the number of characters in *string$*.

See Also `InStr( ), Left$( ), Mid$( ), Right$( ).`

Example

```
Sub Main
   Debug.Print Len("Hello")
End Sub
```

Example Output

```
5
```

Let — Instruction

Syntax **`[Let]`** *`var = expr`*

Description Assign the value of *expr* to *var*. The keyword Let is optional.

Example

```
Sub Main
   Let X = 1
   X = X*2
   Debug.Print X
End Sub
```

Example Output

```
2
```

Like — Operator

Syntax `str1 Like str2`

Description Return the True if str1 matches pattern str2. The pattern in str2 is one or more of the special character sequences shown in the following table.

Char(s)	Description
?	Match any single character.
***	Match zero or more characters.
#	Match a single digit (0-9).
[charlist]	Match any char in the list.
[!charlist]	Match any char not in the list.

Example

```
Sub Main
    Dim X As Object
    Dim Y As Object
    Debug.Print X Is Y ‘ True
End Sub
```

Line Input — Instruction

Syntax **Line Input** [#]*streamnum*, *string$*

Description Get a line of input from *Streamnum* and assign it to *string$*.

See Also Input, Print, Write.

Example

```
Sub Main
   Open “FILENAME.EXT” For Input As #1
   Line Input #1,S$
   Debug.Print S$
   Close #1
End Sub
```

Example Output

ListBox Dialog Item

Definition

Syntax `ListBox x, y, dx, dy, strarray$( ), .field`

Parameters

Name	Description
x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.
y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
strarray$( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
field	The value of the list box is accessed via this *field*. It is the index of the *StrArray$( )* var.

Description Define a listbox item.

See Also `Begin Dialog, Dim As UserDialog.`

Example

```
Sub Main
    Dim lists$(3)
    lists$(0) = “List 0"
    lists$(1) = “List 1"
    lists$(2) = “List 2"
    lists$(3) = “List 3"
    Begin Dialog UserDialog 200,120
        Text 10,10,180,15,"Please push the OK button"
        ListBox 10,25,180,60,lists$(),.list
        OKButton 80,90,40,20
    End Dialog
    Dim dlg As UserDialog
    dlg.list = 2
    Dialog dlg ‘ show dialog (wait for ok)
    Debug.Print dlg.list
End Sub
```

Example Output

Loc

Function

Syntax `Loc(streamnum)`

Parameters

Name	Description
streamnum	Streams 1, 2, 3 and 4 are available in each macro.

Description Return *Streamnum* file position.

Example

```
Sub Main
   Open "FILENAME.EXE" For Input As #1
   L = Loc(1)
   Close #1
   Debug.Print L
End Sub
```

Example Output

```
1
```

Lock

Instruction

Syntax

```
Lock StreamNum
```

-or-

```
Lock StreamNum, RecordNum
```

-or-

```
Lock StreamNum, [start] To end
```

Parameters

Name	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.
start	First record (or byte) in the range.
end	Last record (or byte) in the range.

Description Form 1: Lock all of StreamNum.

Form 2: Lock a record (or byte) of StreamNum.

Form 3: Lock a range of records (or bytes) of StreamNum. If start is omitted then lock starting at the first record (or byte).

Note: Be sure to Unlock for each Lock instruction.

Note: For sequential files (Input, Output and Append) lock always affects the entire file.

See Also `Open, Unlock.`

Example

```
Sub Main
    Dim V As Variant
    Open "SAVE_V.DAT" For Binary As #1
    Lock #1
    Get #1, 1, V
    V = "Hello"
    Put #1, 1, V
    Unlock #1
    Close #1
End Sub
```

LOF — Function

Syntax **Lof(***streamnum***)**

Parameters

Name	Description
streamnum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Description Return StreamNum file length (in bytes).

Example

```
Sub Main
   Open "FILENAME.EXT" For Input As #1
   L = Lof(1)
   Close #1
   Debug.Print L
End Sub
```

Example Output Length of file value.

Log — Function

Syntax `Log(num)`

Parameters

Name	Description
num	Return the natural logarithm of this number value. The value e is approximately 2.718282.

Description Return the natural logarithm.

Example

```
Sub Main
   Debug.Print Log(1)
End Sub
```

Example Output

```
0
```

Log10 — Function

Syntax `Log10(num)`

Parameters

Name	Description
num	Return the base-10 logarithm of this number value.

Description Return the base-10 logarithm.

Example

```
Sub Main
   Debug.Print Log10(24)
End Sub
```

Example Output

```
 1.38021124171161
```

LSet — Instruction

Syntax

```
LSet strvar = str
-or-
LSet usertypevar1 = usertypevar2
```

Description Form 1: Assign the value of *str* to *strvar*. Shorten str by removing trailing chars (or extend with blanks). The previous length *strvar* is maintained.

Form 2: Assign the value of *usertypevar2* to *usertypevar1*. If *usertypevar2* is longer than *usertypevar1* then only copy as much as *usertypevar1* can handle.

See Also `RSet.`

Example

```
Sub Main
    S$ = “123"
    LSet S$ = “A”
    Debug.Print “.”;S$;"."
End Sub
```

Example Output

```
.A  .
```

LTrim$ Function

Syntax **`LTrim`**`[$]`**`(`***`string$`***`)`**

Parameters

Name	Description
string$	Copy this string without the leading spaces.

Description Return the string with *string$*s leading spaces removed.

See Also `Trim$( ), RTrim$( ).`

Example

```
Sub Main
    Debug.Print “.”;LTrim$(“  x  ”);"."
End Sub
```

Example Output

```
.x  .
```

MacroDir$ Function

Syntax **`MacroDir`**`[$]`

Description Return the directory of the current macro. A run-time error occurs if the current macro has never been saved.

See Also MacroRun.

Example

```
Sub Main
   ' Open the file called Data that is in the
   ' same directory as the macro
   Open MacroDir & "\Data" For Input As #1
   Line Input #1, S$
   Close #1
End Sub
```

MacroRun Instruction

Syntax **MacroRun** *command$*

Parameters

Name	Description
command$	Start the macro named by this string value. That macros **Command$** is assigned the text following first space in this value.

Description Play a macro. Execution will continue at the following statement after the macro has completed.

See Also Command$.

Example

```
Sub Main
    Debug.Print "Before Demo"
    MacroRun "APDEMO.APB"
    Debug.Print "After Demo"
End Sub
```

MacroRunThis Instruction

Syntax MacroRunThis MacroCode$

Description Play the macro code. Execution will continue at the following statement after the macro code has completed. The macro code can be either a single line or a complete macro.

Parameter	Description
MacroName$	Run the macro named by this string value.

See Also Command$, MacroDir$, MacroRun.

Example

```
Sub Main
    Debug.Print "Before Demo"
    MacroRunThis "MsgBox ""Hello"""
    Debug.Print "After Demo"
End Sub
```

Main Sub

Syntax

```
Sub Main()
    ...
End Sub
```

-or-

```
Private Sub Main()
    ...
End Sub
```

Description Form 1: Each macro must define Sub Main. A macro is a "program". Running a macro starts the Sub Main and continues to execute until the subroutine finishes.

Form 2: A code module may define a Private Sub Main. This Sub Main is the code module initialization subroutine. If Main is not defined then no special initialization occurs.

See Also **Code Module.**

Mid$ Function/Assignment

Syntax

```
Mid[$](string$, index[, len])
-or-
Mid[$](strvar, index[, len]) = string$
```

Parameters

Name	Description (Mid Function)
string$	Copy chars from this string value.
index	Start copying chars starting at this index value. If the string is not that long then return a null string.
len	Copy this many chars. If the *string$* does not have that many chars starting at *Index* then copy the remainder of *string$*.

Name	Description (Mid Assignment)
strvar	Change part of this string.
index	Change *strvar* starting at this index value. If the string is not that long then it is not changed.
len	The number of chars copied is smallest of: the value of *Len*, the length of *string$* and the remaining length of *strvar*. (If this value is omitted then the number of chars copied is the smallest of: the length of *string$* and the remaining length of *strvar*.)
string$	Copy chars from this string value.

Description

Function: Return the substring of S$ starting at *Index* for *Len* chars.

Instruction: Assign *string$* to the substring in *strvar* starting at *Index* for *Len* chars.

Example

```
Sub Main
    S$ = “Hello There”
    Mid$(S$,7) = “??????????”
    Debug.Print S$ ‘"Hello ?????"
    Debug.Print Mid$(“Hello”,2,1)
End Sub
```

Example Output

```
Hello ?????
e
```

Minute — Function

Syntax `Minute(dateexpr)`

Parameters

Name	Description
dateexpr	Return the minute of the hour for this date value.

Description Return the minute of the hour (0 to 59).

See Also `Hour( ), Second( ), Time( ).`

Example

```
Sub Main
    Debug.Print Minute(#12:15:01 AM#)
End Sub
```

Example Output

```
 15
```

MkDir — Instruction

Syntax `MkDir name$`

Parameters

Name	Description
name$	This string value is the path and name of the directory. A path relative to the current directory can be used.

Description Make directory *name$*.

See Also `RmDir.`

Example

```
Sub Main
   MkDir "C:\APTEMP"
End Sub
```

Month — Function

Syntax `Month(dateexpr)`

Parameters

Name	Description
dateexpr	Return the month of the year for this date value.

Description Return the month of the year (1 to 12).

See Also `Date( ), Day( ), Weekday( ), Year( ).`

Example

```
Sub Main
   Debug.Print Month(#1/1/1900#)
End Sub
```

Example Output

```
 1
```

MsgBox — Instruction/Function

Syntax

```
MsgBox message$[, type][, title$]
-or-
MsgBox(message$[, type][, title$])
```

Parameters

Name	Description
message$	This string value is the text that is shown in the message box.
type	This number value controls the type of message box. See the table below.
title$	This string value is the title of the message box.

Category	Type	Effect (result)
Buttons	0	OK(1) button
	1	OK(1) and Cancel(2) buttons
	2	Abort(3), Retry(4), Ignore(5) buttons
	3	Yes(6), No(7), Cancel(2) buttons
	4	Yes(6) and No(7) buttons
	5	Retry(4) and Cancel(2) buttons
Icons	0	No icon
	16	Stop icon
	32	Question icon
	48	Attention icon
	64	Information icon
Default	0	First button
	256	Second button
	512	Third button
Mode	0	Application modal
	4096	System modal

Description Show a message box titled *Title$*. *Type* controls what the message box looks like (choose one value from each category). Use MsgBox( ) if you need to know what button was pressed. The result indicates which button was pressed.

Example

```
Sub Main
    If MsgBox("Please press OK button",1) = 1 Then
        Debug.Print "OK was pressed"
    Else
        Debug.Print "Cancel was pressed"
    End If
End Sub
```

Name — Instruction

Syntax **Name** *oldname$* **As** *newname$*

Parameters

Name	Description
oldname$	This string value is the path and name of the file. A path relative to the current directory can be used.
newname$	This is the new file name. The file remains in its original directory.

Description Rename file *oldname$* as *newname$*.

Example

```
Sub Main
   Name "AUTOEXEC.BAK" As "AUTOEXEC.SAV"
End Sub
```

Now — Function

Syntax **Now**

Description Return the current date and time as a *date* value.

See Also Date, Time, Timer.

Example

```
Sub Main
   Debug.Print Now
```

```
End Sub
```

Example Output

```
2/9/96 7:59:26 AM
```

Oct$ Function

Syntax `Oct[$](num)`

Parameters

Name	Description
num	Return an octal encoded string for this number value.

Description Return a octal string.

See Also `Hex$( ), Str$( ), Val( ).`

Example

```
Sub Main
   Debug.Print Oct$(15)
End Sub
```

Example Output

```
17
```

Object Module

Description (The Object module feature is not implemented in version 1.5 of APWIN Basic)

An object module implements an OLE Automation object.

o It has a set of Public properties, functions and subroutines accessible from other macros and modules.

o These public symbols are accessed via the name of the object module or an object variable.

o Public Consts, Types, arrays, fixed length strings are not allowed.

o An object module is similar to a class module except that one instance is automatically created. That instance has the same name as the object module's name.

o To create additional instances use:

```
Dim Obj As objectname
Set Obj = New objectname
```

See Also Class Module, Code Module, Uses.

Example

```
'A.WWB
'#Uses "System.OBM"
Sub Main
    Debug.Print Hex(System.Version)
End Sub

'System.OBM
Option Explicit
Declare Function GetVersion16 Lib "Kernel" _
    Alias "GetVersion" () As Long
Declare Function GetVersion32 Lib "Kernel32" _
    Alias "GetVersion" () As Long

Public Function Version() As Long
    If Win16 Then
        Version = GetVersion16
    Else
        Version = GetVersion32
    End If
End Function
```

Object_Initialize Sub

Syntax

```
Private Sub Object_Initialize()
    ...
End Sub
```

Description Object module initialization subroutine. Each time a new instance is created for a Object module the Object_Initialize sub is called. If Object_Initialize is not defined then no special initialization occurs.

Note: Object_Initialize is also called for the instance that is automatically created.

See Also

Object Module, Object_Terminate.

Object_Terminate Sub

Syntax

```
Private Sub Object_Terminate()
    ...
End Sub
```

Description Object module termination subroutine. Each time an instance is destroyed for a Object module the Object_Terminate sub is called. If Object_Terminate is not defined then no special termination occurs.

See Also Object Module, Object_Initialize.

Oct$ Function

Syntax **Oct[$](*Num*)**

Description Return a octal string.

Parameter	Description
Num	Return an octal encoded string for this number value.

See Also Hex$( ), Str$( ), Val( ).

Example

```
Sub Main
    Debug.Print Oct$(15) '17
End Sub
```

OKButton Dialog Item Definition

Syntax **OKButton** x, y, dx, dy[, *.field*]

Parameters

Name	Description

x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.
y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
field	This identifier is the name of the *field*. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used. If this is omitted then the field name is OK.

Description Define an OK button item. Pressing the OK button updates the *dlgvar* field values and closes the dialog. (**Dialog**( ) function call returns -1.)

See Also `Begin Dialog, Dim As` *`UserDialog`*.

Example

```
Sub Main
    Begin Dialog UserDialog 200,120
        Text 10,10,180,30,"Please push the OK button"
        OKButton 80,90,40,20
    End Dialog
    Dim dlg As UserDialog
    Dialog dlg    'Show dialog (Wait for OK)
End Sub
```

On Error — Instruction

Syntax

```
On Error GoTo 0
-or-
On Error GoTo label
-or-
On Error Resume Next
```

Description Form 1: Disable the error handler (default).

Form 2: Send error conditions to an error handler.

Form 3: Error conditions continue execution at the next statement.

On Error sets or disables the error handler. Each user defined subroutine, function or property has its own error handler. The default is to terminate the macro on any error. The **Err** variable is set whenever an error occurs. Once an error has occurred and the error handler is executing any further errors will terminate the macro, unless **Err** has been set to zero.

Note: This instruction resets **Err** to zero and **Error$** to null.

Example

```
Sub Main
    On Error Resume Next
    Error 1
    Debug.Print "RESUMING, Err=";Err
    On Error GoTo X
    Error 1
    Exit Sub

X:  Debug.Print "Err=";Err
    Err = 0
    Resume Next
End Sub
```

Example Output

```
RESUMING, Err= 1
Err= 1
```

Open

Instruction

Syntax **Open** *name$* **For** *mode* **As** [#]*streamnum*

Parameters

Name	Description
name$	This string value is the path and name of the file. A path relative to the current directory can be used.
mode	May be Input, Output or Append.
streamnum	Streams 1, 2, 3 and 4 are available in each macro.

Description Open file *Name$* for mode as *Streamnum*.

See Also Close, Reset.

Example

```
Sub Main
Open "FILENAME.EXT" For Output As #1
   Print #1,"1,2,""Hello"""
   Close #1
End Sub
```

Option

Definition

Syntax `Option Explicit`

Description Require all variables to be declared prior to use. Variables are declared using **Dim**, **Private** or **Public** or **Static**.

See Also Option Explicit

Example

```
Option Explicit
Sub Main
    Dim A
    A = 1
    B = 2   'B has not been declared.
End Sub
```

OptionButton Dialog Item

Definition

Syntax **OptionButton** *x*, *y*, *dx*, *dy*, *title$*[, *.field*]

Parameters

Name	Description
x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.
y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
title$	The value of this string is the title of the option button.

Description Define an option button item.

See Also `Begin Dialog, Dim As` *`UserDialog`*`, OptionGroup.`

Example

```
Sub Main
    Begin Dialog UserDialog 200,120
        Text 10,10,180,15,"Please push the OK button."
        OptionGroup .options
            OptionButton 10,30,180,15,"Option &0"
            OptionButton 10,45,180,15,"Option &1"
            OptionButton 10,60,180,15,"Option &2"
        OKButton 80,90,40,20
    End Dialog
    Dim dlg As UserDialog
    dlg.options = 2
    Dialog dlg              'Show dialog (Wait for OK)
    Debug.Print dlg.options
End Sub
```

OptionGroup Dialog Item

Definition

Syntax

OptionGroup .*field*

OptionButton *x*, *y*, *dx*, *dy*, *title$*[, *.field*]

OptionButton *x*, *y*, *dx*, *dy*, *title$*[, *.field*]

...

Parameters

Name	Description
field	The value of the option group is accessed via this field. This first option button is 0, the second is 1, etc.
x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.
y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
title$	The value of this string is the title of the option button.

Description Define a optiongroup and option button items.

See Also `Begin Dialog, Dim As UserDialog, OptionButton.`

Example

```
Sub Main
    Begin Dialog UserDialog 200,120
        Text 10,10,180,15,"Please push the OK button."
        OptionGroup .options
            OptionButton 10,30,180,15,"Option &0"
            OptionButton 10,45,180,15,"Option &1"
            OptionButton 10,60,180,15,"Option &2"
        OKButton 80,90,40,20
    End Dialog
    Dim dlg As UserDialog
    dlg.options = 2
    Dialog dlg          'Show dialog (Wait for OK)
    Debug.Print dlg.options
End Sub
```

Pow — Function

Syntax **Pow(***numx, powery***)**

Parameters

Name	Description
numx	Number X to be rased.
powery	Power of Y.

Description Return the value of a number (NumX) raised to the power of (PowerY).

Example

```
Sub Main
   Debug.Print Pow(3,3)
End Sub
```

Example Output

```
 27
```

Picture Dialog Item

Definition

Syntax

```
Picture X, Y, DX, DY, FileName$, Type[, .Field]
```

Description

Define a picture item. The bitmap is automatically sized to fit the item's entire area.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
FileName$	The value of this string is the .BMP file shown in the picture control.
Type	This numeric value indicates the type of bitmap used. See below.
Field	This identifier is the name of the field. The dialogfunc receives this name as string. If this identifer is omitted then the first two words of the title are used.

Type	Effect
0	FileName is the name of the bitmap file. If the file does not exist then "(missing picture)" is displayed.
3	The clipboard's bitmap is displayed. Not supported.
+16	Instead of displaying "(missing picture)" a run-time error occurs.

See Also

Begin Dialog, Dim As UserDialog.

Example

```
Sub Main
    Begin Dialog UserDialog 200,120
        Picture 10,10,180,75,"SAMPLE.BMP",0
        OKButton 80,90,40,20
    End Dialog
    Dim dlg As UserDialog
```

```
    Dialog dlg ‘ show dialog (wait for ok)
End Sub
```

PowerRatioTodB — Function

Syntax **PowerRatioTodB(***num***)**

Parameters

Name	Description
num	

Description Return the value in dB of the power ratio of *num* to 1.

Example

```
Sub Main
   Debug.Print Format(PowerRatioTodB(.5), “#.0000")
End Sub
```

Example Output

```
 -3.0103
```

Equation

```
PowerRatioTodB = 10 * Log10(Num)
```

Print — Instruction

Syntax **Print** *#streamnum*, [*expr*[; ...][;]]

Description Print the *expr*(s) to *Streamnum*. Use ; to separate expressions. A *num* is automatically converted to a string before printing (just like **Str$**( )). If the instruction does not end with a ; then a newline is printed at the end.

See Also Input, Line Input, Write.

Example

```
Sub Main
   A = 1
   B = 2
   C$ = Hello
   Open “FILENAME.EXT” For Output As #1
   Print #1,A;",";B;",""";C$;"""
   Close #1
End Sub
```

Private — Definition

Syntax ***Private*** *name*[*type*][([Dim[, …]])] [As *type*][, ...]

Description Create arrays (or simple variables) which are available to the entire macro, but not other macros. Dimension var array(s) using the *dimlist* to establish the minimum and maximum index value for each dimension. If the *dims* is omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dims*. It must be **ReDim**ensioned before it can be used. The Private statement must be placed outside of **Sub**, **Function** or **Property** blocks.

See Also Dim, Public, ReDim, Static.

Example

```
Private A0,A1(1),A2(1,1)
Sub Init
   A0 = 1
   A1(0) = 2
   A2(0,0) = 3
End Sub
Sub Main
   Init
   Debug.Print A0;A1(0);A2(0,0)
End Sub
```

Example Output

```
 1 2 3
```

Private — Keyword

Description Private **Consts, Declares, Functions, Privates, Propertys, Subs** and **Types** are only available in the current macro.

Property — Definition

Syntax [Private|Public] **Property Get** *name*[*type*][([*param*[, _

```
  ]])] [As type]
  statements
End Property

-or-

[Private|Public] Property [LetSet] name[([param[, _
  ]])]
  statements
End Property
```

Description

User defined property. The property defines a set of *statements* to be executed when its value is used or changed. A property acts like a variable, except that getting its value calls Property Get and changing its value calls Property Let (or Property Set). Property Get and Property Let with the same *name* define a property that holds a value. Property Get and Property Set with the same *name* define a property that holds an object reference. The values of the calling *arglist* are assigned to the parameters in the *params*.

For Property Let and Property Set the last parameter is the value on the right hand side of the assignment operator.

Public is assumed if neither *Private* or *Public* is specified.

See Also

`Function, Sub.`

Example

```
Dim X_Value
Property Get X()
    X = X_Value
End Property
Property Let X(NewValue)
    If Not IsNull(NewValue) Then X_Value = NewValue
End Property

Sub Main
    X = “Hello”
    Debug.Print X
    X = Null
    Debug.Print X
End Sub
```

Example Output

```
Hello
Null
```

Public — Definition

Syntax ***Public*** *name*[*type*][([Dim[, …]])] [As *type*][, ...]

Description Create arrays (or simple variables) which are available to the entire macro and other macros. Dimension var array(s) using the *dims* to establish the minimum and maximum index value for each dimension. If the *dims* are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dims*. It must be **ReDim**ensioned before it can be used. The Public statement must be placed outside of **Sub**, **Function** or **Property** blocks.

See Also Dim, Private, ReDim, Static.

Example

```
Public A0,A1(1),A2(1,1)
   Sub Init
   A0 = 1
   A1(0) = 2
   A2(0,0) = 3
End Sub
Sub Main
   Init
   Debug.Print A0;A1(0);A2(0,0)
End Sub
```

Example Output

```
 1 2 3
```

Public — Keyword

Description Public **Consts**, **Declares, Functions, Propertys, Publics, Subs** and **Types** in hidden macros are available in all other macros.

PushButton Dialog Item — Definition

Syntax **PushButton** *x*, *y*, *dx*, *dy*, *title$*[, *.field*]

Parameters

Name	Description

x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.
y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
title$	The value of this string is the title of the push button control.
field	This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used.

Description Define a push button item. Pressing the push button updates the *dlgvar* field values and closes the dialog. (**Dialog**( ) function call returns the push buttons ordinal number in the dialog. The first push button returns 1.)

See Also `Begin Dialog, Dim As UserDialog.`

Example

```
Sub Main
    Begin Dialog UserDialog 200,120
        Text 10,10,180,30,"Please push the DoIt button"
        OKButton 40,90,40,20
        PushButton 110,90,60,20,"&Do It"
    End Dialog
    Dim dlg As UserDialog
    Debug.Print Dialog(dlg)
End Sub
```

Put — Instruction

Syntax **Put** *StreamNum*, [*RecordNum*], *var*

Parameters

Name	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1. If this is omitted then the current position (or record number) is used.
var	This variable value is written to the file. For a fixed length variable (like Long) the number of bytes required to store the variable are written. For a Variant variable two bytes which describe its type are written and then the variable value is written accordingly. For a usertype variable each field is written in sequence. For an array variable each element is written in sequence. For a dynamic array variable the number of dimensions and range of each dimension is written prior to writing the array values. All binary data values are written to the file in little-endian format.

Note: When a writing string (or a dynamic array) to a Binary mode file the string length (or array dimension) information is not written. Only the string data or array elements are written.

Description Write a variable's value to StreamNum.

See Also `Get, Open.`

Example

```
Sub Main
    Dim V As Variant
    Open "SAVE_V.DAT" For Binary Access Write As #1
    Put #1, , V
    Close #1
End Sub
```

QBColor — Function

Syntax **QBColor(***num***)**

Parameters

num	color
0	black
1	blue
2	green
3	cyan
4	red

5	magenta
6	yellow
7	white
8	gray
9	light blue
10	light green
11	light cyan
12	light red
13	light magenta
14	light yellow
15	bright white

Description Return the appropriate color defined by Quick Basic.

See Also `RGB( ).`

Example

```
Sub Main
   Debug.Print Hex(QBColor(1))
   Debug.Print Hex(QBColor(7))
   Debug.Print Hex(QBColor(8))
   Debug.Print Hex(QBColor(9))
   Debug.Print Hex(QBColor(10))
   Debug.Print Hex(QBColor(12))
   Debug.Print Hex(QBColor(15))
End Sub
```

Example Output

```
800000
C4C4C4
808080
FF0000
FF00
FF
FFFFFF
```

Randomize — Instruction

Syntax **`Randomize`**

Description Randomize the random number generator.

See Also Rnd( ).

Example

```
Sub Main
   Randomize
   Debug.Print Rnd
End Sub
```

Example Output

```
0.84881130405591
```

ReDim — Instruction

Syntax **ReDim** [Preserve] *name*[*type*][([Dim[, …]])] [As _ *type*][, ...]

Description Redimension a dynamic array. Use Preserve to keep the array values. Otherwise, the array values will all be reset. When using **Preserve** only the last index of the array may change. The number of indexes may not. (A one-dimensional array cant be redimensioned as a two-dimensional array.)

See Also Dim, Private, Public, Static.

Example

```
Sub Main
   Dim X()
   ReDim X(3)
   Debug.Print UBound(X)
   ReDim X(200)
   Debug.Print UBound(X)
End Sub
```

Example Output

```
3
200
```

Reference — Comment

Syntax **'#Reference**
{*uuid***}#***vermajor.verminor*#*lcid*#**[***path***[#***name***]]**

Description The Reference comment indicates that the current macro/module references the type library identified. Reference comment lines must be the first lines in the macro/module (following the global Attributes). Reference comments are in reverse priority (from lowest to highest). The IDE does not display the reference comments.

Parameters

Name	Description
uuid	Type library's universally unique identifier.
vermajor	Type library's major version number.
verminor	Type library's minor version number.
lcid	Type library's locale identifier.
path	Type library's path.
name	Type library's name.

Example

```
'#Reference
{00025E01-0000-0000-C000-000000000046}#4.0#0#C: _
\PROGRAM FILES\COMMON FILES\MICROSOFT SHARED\DAO\_
DAO350.DLL#Microsoft DAO 3.5 Object Library
```

Rem

Instruction

Syntax

```
Rem ...
-or-
'...
```

Description Both forms are comments. The Rem form is an instruction. The ' form can be used at the end of any macro line. All text from either " ' " or Rem to the end of the line is part of the comment. That text is not executed.

Example

```
Sub Main
    Debug.Print "Hello" 'Prints to the output window.
    Rem the macro terminates at Main's End Sub
End Sub
```

Example Output

```
Hello
```

Replace — Function

Syntax `Replace[$](S, Pat, Rep, [Index], [Count])`

Description Replace Pat with Rep in S.

Parameters

Name	Description
S	This string value is searched. Replacements are made in the string returned by Replace.
Pat	This string value is the pattern to look for.
Rep	This string value is the replacement.
Index	This numeric value is the starting index in S. Replace(S,Pat,Rep,N) is equivalent to Replace(Mid(S,N),Pat,Rep). If this is omitted use 1.
Count	This numeric value is the maximum number of replacements that will be done. If this is omitted use -1 (which means replace all occurrences).

See Also `InStr( ), InStrRev( ), Left$( ), Len( ), Mid$( ), Right$( ).`

Example

```
Sub Main
    Debug.Print Replace$(“abcabc”,"b","B")     ‘"aBcaBc"
    Debug.Print Replace$(“abcabc”,"b","B",,1) ‘"aBcabc"
    Debug.Print Replace$(“abcabc”,"b","B",3)  ‘"caBc"
    Debug.Print Replace$(“abcabc”,"b","B",9)  ‘""
End Sub
```

Reset — Instruction

Syntax `Reset`

Description Close all open streams for the current macro.

See Also `Close, Open.`

Example

```
Sub Main
    ‘ Read the first line of XXX and print it.
    Open “FILENAME.EXT” For Input As #1
    Line Input #1,L$
```

```
    Debug.Print L$
    Reset
End Sub
```

Resume — Instruction

Syntax

```
Resume label
-or-
Resume Next
```

Description

Form 1: Resume execution at *label*.

Form 2: Resume execution at the next statement.

Once an error has occurred, the error handler can use Resume to continue execution. The error handler must use Resume or **Exit** at the end. Executing an End **Sub** (or End **Function**) while in an error handler causes a run-time error.

Note: This instruction resets **Err** to zero and **Error$** to null.

Example

```
Sub Main
    On Error GoTo X
    Error 1
    Debug.Print "RESUMING"
    Exit Sub

X:  Debug.Print "Err=";Err
    Resume Next
End Sub
```

Example Output

```
RESUMING
```

RGB — Function

Syntax

```
RGB(red, green, blue)
```

Description

Return a color.

See Also

```
QBColor( ).
```

Example

```
Sub Main
   Debug.Print Hex(RGB(255,0,0))
End Sub
```

Example Output FF

Right$ Function

Syntax **Right**[$]**(***string$*, *len***)**

Parameters

Name	Description
string$	Return the right portion of this string value.
len	Return this many chars. If *string$* is shorter than that then just return *string$*.

Description Return the last Len chars of *string$*.

See Also InStr( ), Left$( ), Len( ), Mid$( ).

Example

```
Sub Main
   Debug.Print Right$(Hello,3)
End Sub
```

Example Output Llo

RmDir — Instruction

Syntax `RmDir name$`

Parameters

Name	Description
name$	This string value is the path and name of the directory. A path relative to the current directory can be used.

Description Remove directory Name$.

See Also `MkDir.`

Example

```
Sub Main
   RmDir "C:\APTEMP"
End Sub
```

Rnd — Function

Syntax `Rnd([num])`

Parameters

Name	Description
num	This number value is ignored.

Description Return a random number greater than or equal to zero and less than one.

See Also `Randomize.`

Example

```
Sub Main
   Debug.Print Rnd()
End Sub
```

Example Output

```
 0.95883053071688
```

RSet — Instruction

Syntax **RSet** *strvar = str*

Description Assign the value of *str* to *strvar*. Shorten *str* by removing trailing chars (or extend with leading blanks). The previous length *strvar* is maintained.

See Also LSet.

Example

```
Sub Main
    S$ = "123"
    RSet S$ = "A"
    Debug.Print ".";S$;"."
End Sub
```

Example Output . A.

RTrim$ — Function

Syntax **RTrim**[$]**(***string$***)**

Parameters

Name	Description
string$	Copy this string without the trailing spaces.

Description Return the string with *string$*s trailing spaces removed.

See Also LTrim$( ), Trim$( ).

Example

```
Sub Main
    Debug.Print ".";RTrim$("  x  ");"."
End Sub
```

Example Output . x.

SaveSetting — Instruction

Syntax **SaveSetting** *AppName$, Section$, Key$, Setting*

Description Save the Setting for Key in Section in project AppName. Win16 and Win32s store settings in a .ini file named AppName. Win32 stores settings in the registration database.

Parameter	Description
AppName$	This string value is the name of the project which has this Section and Key.
Section$	This string value is the name of the section of the project settings.
Key$	This string value is the name of the key in the section of the project settings.
Setting	Set the key to this value. (The value is stored as a string.)

Example

```
Sub Main
    SaveSetting “MyApp”,"Font","Size",10
End Sub
```

Second — Function

Syntax **Second(***dateexpr***)**

Parameters

Name	Description
dateexpr	Return the second of the minute for this date value.

Description Return the second of the minute (0 to 59).

See Also Hour( ), Minute( ), Time( ).

Example

```
Sub Main
    Debug.Print Second(#12:00:01 AM#)
End Sub
```

Example Output

```
 1
```

Seek — Instruction

Syntax **Seek** [#]*streamnum*, *count*

Parameters

Name	Description
streamnum	Streams 1, 2, 3 and 4 are available in each macro.
count	This number value is the number of bytes to skip over from the beginning of the file.

Description Position *Streamnum* for input *Count*.

See Also Seek( ).

Example

```
Sub Main
    Open “FILEMANE.EXT” For Input As #1
    Line Input #1,L$
    Seek #1,0 ‘ Rewind to start of file.
    Input #1,A
    Close #1
    Debug.Print A
End Sub
```

Seek — Function

Syntax **Seek(***streamnum***)**

Parameters

Name	Description
streamnum	Streams 1, 2, 3 and 4 are available in each macro.

Description Return StreamNum current position.

See Also Seek.

Example

```
Sub Main
    Open “FILENAME.EXT” For Input As #1
    Line Input #1,L$
    Debug.Print Seek(1)
    Close #1
End Sub
```

Select Case Statement

Syntax

```
Select Case expr
  Case caseexpr[, ...]
        statements
  [Case Else
        statements]
End Select
```

Parameters

caseexpr	Description
expr	Execute if equal.
Is < expr	Execute if less than.
Is <= expr	Execute if less than or equal to.
Is > expr	Execute if greater than.
Is >= expr	Execute if greater than or equal to.
Is <> expr	Execute if not equal to.
expr1 To expr2	Execute if greater than or equal to *expr1* and less than or equal to *expr2*.

Description

Select the appropriate case by comparing the *expr* with each of the caseexprs. Select the Case Else part if no caseexpr matches. (If the Case Else is omitted then skip the entire Select...End Select block.)

See Also

If, Choose( ), IIf( ).

Example

```
Sub Main
    S$ = InputBox$("Enter hello, goodbye, dinner or
sleep:")
    Select Case UCase$(S$)
    Case "HELLO"
        Debug.Print "come in"
    Case "GOODBYE"
        Debug.Print "see you later"
    Case "DINNER"
        Debug.Print "Please come in."
        Debug.Print "Dinner will be ready soon."
    Case "SLEEP"
        Debug.Print "Sorry."
        Debug.Print "We are full for the night"
```

```
        Case Else
            Debug.Print "What?"
        End Select
    End Sub
```

Example Output

SendKeys — Instruction

Syntax **SendKeys** *keys$*[, *wait*]

Parameters

Name	Description
keys$	Send the keys in this string value to Windows.
wait	If this is not zero then the keys are sent before executing the next instruction. If this is omitted or zero then the keys are sent during the following instructions.

Keys$	Description
+	Shift modifier key: the following key is a shifted key
^	Ctrl modifier key: the following key is a control key
%	Alt modifier key: the following key is an alt key
~	Enter key
(keys)	Modifiers apply to all keys
{special n}	special key (n is an optional repeat count)
k	k Key (k is any single char)
K	Shift k Key (K is any capital letter)

Description Send *Keys$* to Windows.

Special Keys:

Key	Description
k	k Key (any single char)
Cancel	Break Key
Esc or	
Escape	Escape Key
Enter	Enter Key
Menu	Menu Key (Alt)
Help	Help Key (?)

Prtsc	Print Screen Key
Print	?
Select	?
Execute	?
Tab	Tab Key
Pause	Pause Key
BS, BkSp or BackSpace	Back Space Key
Del or Delete	Delete Key
Ins or Insert	Insert Key
K	shift k Key
Left	Left Arrow Key
Right	Right Arrow Key
Up	Up Arrow Key
Down	Down Arrow Key
PgUp	Page Up Key
PgDn	Page Down Key
Home	Home Key
End	End Key
Clear	Num Pad 5 Key
Pad0 to Pad9	Num Pad 0 to 9 Keys
*Pad**	Num Pad * Key
Pad+	Num Pad + Key
PadEnter	Num Pad Enter Key
Pad-	Num Pad - Key
Pad.	Num Pad . Key
Pad/	Num Pad / Key
F1 to F24	F1 to F24 Keys

See Also AppActivate, Shell( ).

Example

```
Sub Main
    SendKeys “%S”     ‘ send Alt-S (Search)
    SendKeys “GoTo~~” ‘ send G o T o {Enter} {Enter}
End Sub
```

Set — Instruction

Syntax

```
Set objvar = objexpr
-or-
Set objvar = New objtype
```

Description

Form 1: Set *objvar*s object reference to the object reference of *objexpr*.

Form 2: Set *objvars* object reference to the a new instance of *cotype* (a component object type.)

The Set instruction is how object references are assigned.

Example

```
Sub Main
   Dim Excel As Object
   Set Excel = CreateObject("Excel.Application")
End Sub
```

SetAttr — Instruction

Syntax

```
SetAttr name$, attrib
```

Parameters

Name	Description
name$	This string value is the path and name of the file. A path relative to the current directory can be used.
attrib	Set the files *attributes* to this numeric value.

Description

Set the *attributes* for file *Name$*. If the file does not exist then a run-time error occurs.

Example

```
Sub Main
    Attrib = GetAttr("FILENAME.EXT")
    SetAttr "FILENAME.EXE",1  'Readonly
    Debug.Print GetAttr("FILENAME.EXE")
    SetAttr "FILENAME.EXE",Attrib
End Sub
```

Example Output

```
1
```

Sgn — Function

Syntax `Sgn(num)`

Parameters

Name	Description
num	Return the sign of this number value. Return -1 for negative. Return 0 for zero. Return 1 for positive.

Description Return the sign.

Example

```
Sub Main
   Debug.Print Sgn(9)
   Debug.Print Sgn(0)
   Debug.Print Sgn(-9)
End Sub
```

Example Output

```
 1
 0
-1
```

Shell — Function

Syntax `Shell(name$[, windowtype])`

Parameters

Name	Description
name$	This string value is the path and name of the program to run. Command line arguments follow the program name. (A long file name containing a space must be surrounded by literal double quotes.)
windowtype	This controls how the applications main window is shown. See the table below.

WindowType	Effect
1, 5, 9	Normal Window
2	Minimized Window (default)
3	Maximized Window
4, 8	Normal Deactivated Window
6, 7	Minimized Deactivated Window

Description Execute program *Name$*. This is the same as using File|Run from the Program Manager. This instruction can run .COM, .EXE, .BAT and .PIF files. If successful, return the task ID.

See Also `AppActivate, SendKeys.`

Example

```
Sub Main
    X = Shell("Calc",4) 'Run the calc program.
    AppActivate "Calculator"
    SendKeys "10{+}30*2=",1 '70
End Sub
```

Sin — Function

Syntax **`Sin(`***`num`***`)`**

Parameters

Name	Description
num	Return the sine of this number value. This is the number of radians. There are 2*Pi radians in a full circle.

Description Return the sine.

Example

```
Sub Main
   Debug.Print Sin(1)
End Sub
```

Example Output

```
0.841470984807897
```

Space$ — Function

Syntax **`Space[$](`***`len`***`)`**

Parameters

Name	Description
len	Create a string this many spaces long.

Description Return the string *Len* spaces long.

See Also `String$( ).`

Example

```
Sub Main
```

```
    Debug.Print "."; Space$(3); "."
End Sub
```

Example Output

```
.    .
```

Sqr — Function

Syntax `Sqr(num)`

Parameters

Name	Description
num	Return the square root of this number value.

Description Return the square root.

Example

```
Sub Main
   Debug.Print Sqr(9)
End Sub
```

Example Output

```
3
```

Static — Definition

Syntax `Static name[type][([Dim[, …]])] [As type][, ...]`

Description A static variable retains it value between procedure calls. Dimension var array(s) using the *dims* to establish the minimum and maximum index value for each dimension. If the *dims* is omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dims*. It must be **ReDim**ensioned before it can be used.

See Also `Dim, Private, Public, ReDim.`

Example

```
Sub A
    Static X
    Debug.Print X
    X = "Hello"
End Sub

Sub Main
```

```
    A
    A ' prints "Hello"
End Sub
```

Example Output

```
Hello
```

Stop — Instruction

Syntax `Stop`

Description Pause macro execution. If execution is resumed then it starts at the next instruction. Use **End** to terminate the macro completely.

Example

```
Sub Main
    For I = 1 To 10
        Debug.Print I
        If I = 3 Then Stop
    Next I
End Sub
```

Example Output

```
1
2
3
```

Str$ — Function

Syntax `Str[$](num)`

Parameters

Name	Description
Len	Return the string representation of this number value. Positive values begin with a blank. Negative values begin with a dash -.

Description Return the string representation of *num*.

See Also `CStr( ), Hex$( ), Oct$( ), Val( ).`

Example

```
Sub Main
   Debug.Print Str$(9*9)
```

```
End Sub
```

Example Output

```
81
```

StrComp$ Function

Syntax **StrComp(***Str1*,*Str2*,*Comp*)

Description Compare two strings.

Parameter	Description
Str1	Compare this string with Str2. If this value is Null then Null is returned.
Str2	Compare this string with Str1. If this value is Null then Null is returned.
Comp	This numeric value indicates the type of comparison. If this is omitted or zero then binary comparison is used. Otherwise, text comparison is used. (Text comparison is not case sensitive.)

Result	Description
-1	Str1 is less than Str2.
0	Str1 is equal to Str2.
1	Str1 is greater than Str2.
Null	Str1 or Str2 is Null.

See Also LCase$( ), StrConv$( ), UCase$( ).

Example

```
Sub Main
    Debug.Print StrComp(“F”,"e") ‘ -1
    Debug.Print StrComp(“F”,"e",1) ‘ 1
    Debug.Print StrComp(“F”,"f",1) ‘ 0
End Sub
```

StrConv$ Function

Syntax **StrConv[*$*](*Str*,*Conv*)**

Description Convert the string.

Parameter	Description
Str	Convert this string value. If this value is Null then Null is returned.
Conv	This numeric value indicates the type of conversion. See conversion table below.

Conv	Value	Effect
vbUpperCase	*1*	Convert to upper case.
vbLowerCase	*2*	Convert to lower case.
vbProperCase	*3*	Convert to proper case. (Not supported.)
vbWide	*4*	Convert to wide. (Only supported for Win32 in eastern locales.)
vbNarrow	*8*	Convert to narrow. (Only supported for Win32 in eastern locales.)
vbKatakana	*16*	Convert to Katakana. (Only supported for Win32 in Japanese locales.)
vbHiragana	*32*	Convert to Hiragana. (Only supported for Win32 in Japanese locales.)
vbUnicode	*64*	Convert to Unicode. (Only supported for Win32.)
vbFromUnicode	*128*	Convert from Unicode. (Only supported for Win32.)

See Also LCase$( ), StrComp( ), UCase$( ).

Example

```
Sub Main
    Dim B(1 To 3) As Byte
    B(1) = 65
    B(2) = 66
    B(3) = 67
    Debug.Print StrConv$(B,vbUnicode) '"ABC"
End Sub
```

String$ Function

Syntax **String**[$]**(***len, CHAR|$***)**

Parameters

Name	Description
len	Create a string this many chars long.
char|$	Fill the string with this char value. If this is a number value then use the ASCII char equivalent. If this is a string value use the first char of that string.

Description Return the string *Len* long filled with *Char* or the first char of *Char$*.

See Also `Space$( ).`

Example

```
Sub Main
   Debug.Print String$(4,65)
   Debug.Print String$(4,"ABC")
End Sub
```

Example Output

```
AAAA
AAAA
```

Sub

Definition

Syntax

```
[Private|Public] Sub name[([param[, …]])]
  statements
End Sub
```

Description User defined subroutine. The subroutine defines a set of *statements* to be executed when it is called. The values of the calling *arglist* are assigned to the *params*. A subroutine does not return a result. Every macro has at least one subroutine. Sub Main must be defined. The macros execution begins at Sub Main. Sub Main must not have any *params*.

Public is assumed if neither *Private* or *Public* is specified.

See Also `Declare, Function, Property.`

Example

```
Sub IdentityArray(A()) ' A() is an array of numbers
    For I = LBound(A) To UBound(A)
        A(I) = I
    Next I
End Sub
```

```
Sub CalcArray(A(),B,C) ‘ A() is an array of numbers
    For I = LBound(A) To UBound(A)
        A(I) = A(I)*B+C
    Next I
End Sub

Sub ShowArray(A()) ‘ A() is an array of numbers
    For I = LBound(A) To UBound(A)
        Debug.Print “(”;I;")=";A(I)
    Next I
End Sub

Sub Main
    Dim X(1 To 4)
    IdentityArray X() ‘ X(1)=1, X(2)=2, X(3)=3, X(4)=4
    CalcArray X(),2,3 ‘ X(1)=5, X(2)=7, X(3)=9, X(4)=11
    ShowArray X()     ‘ print X(1), X(2), X(3), X(4)
End Sub
```

Example Output

```
( 1)= 5
( 2)= 7
( 3)= 9
( 4)= 11
```

Tan — Function

Syntax **Tan(***num***)**

Parameters

Name	Description
num	Return the tangent of this number value.

Description Return the tangent.

Example

```
Sub Main
   Debug.Print Tan(1)
End Sub
```

Example Output

```
1.5574077246549
```

Text Dialog Item

Definition

Syntax

```
Text x, y, dx, dy, title$[, .field]
```

Parameters

Name	Description
x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.
y	This number value is the distance from the top edge of the dialog box. It is measure d in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
title$	The value of this string is the title of the text control.
field	This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used.

Description

Define a text item.

See Also

Begin Dialog, Dim As *UserDialog*.

Example

```
Sub Main
    Begin Dialog UserDialog 200,120
        Text 10,10,180,15,"Please push the OK button."
        OKButton 80,90,40,20
    End Dialog
    Dim dlg As UserDialog
    Dialog dlg       'Show dialog (Wait for OK)
End Sub
```

TextBox Dialog Item

Definition

Syntax

```
TextBox x, y, dx, dy, .field$[, options]
```

Parameters

Name	Description

x	This number value is the distance from the left edge of the dialog box. It is measured in 1/8ths of the average character width for the dialogs font.
y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12ths of the character height for the dialogs font.
dx	This number value is the width. It is measured in 1/8ths of the average character width for the dialogs font.
dy	This number value is the height. It is measured in 1/12ths of the character height for the dialogs font.
field	The value of the text box is accessed via this field.
options	If this numeric value is zero or omitted then a single line of text can be entered. If it is less than zero then a hidden password can be entered. If it is greater than zero then multiple lines of text can be entered.

Description Define a textbox item.

See Also `Begin Dialog, Dim As UserDialog.`

Example

```
Sub Main
    Begin Dialog UserDialog 200,120
        Text 10,10,180,15,"Please push the OK button"
        TextBox 10,25,180,20,.Text$
        OKButton 80,90,40,20
    End Dialog
    Dim dlg As UserDialog
    dlg.Text$ = “none”
    Dialog dlg ‘ show dialog (wait for ok)
    Debug.Print dlg.Text$
End Sub
```

Time — Function

Syntax `Time[$]`

Description Return the current time as a *date* value.

See Also `Date, Now, Timer.`

Example

```
Sub Main
```

```
   Debug.Print Time
End Sub
```

Example Output `12:04:25 PM`

Timer — Function

Syntax **`Timer`**

Description Return the number of seconds past midnight. (This is a real number, accurate to about 1/18th of a second.)

See Also `Date, Now, Time.`

Example

```
Sub Main
   Debug.Print Timer
End Sub
```

Example Output `45284.53`

TimeSerial — Function

Syntax **`TimeSerial(`***`hour, minute, second`***`)`**

Parameters

Name	Description
hour	This numeric value is the hour (0 to 23).
minute	This numeric value is the minute (0 to 59).
second	This numeric value is the second (0 to 59).

Description Return a *date* value.

See Also `DateSerial, DateValue, TimeValue.`

Example

```
Sub Main
   Debug.Print TimeSerial(13,30,0)
End Sub
```

Example Output `1:30:00 PM`

TimeValue — Function

Syntax `TimeValue(date$)`

Parameters

Name	Description
date$	Convert this string value to the time part of date it represents.

Description Return the time part of date encoded as a string value.

See Also `DateSerial, DateValue, TimeSerial.`

Example

```
Sub Main
    Debug.Print TimeValue("1/1/2000 12:00:01 AM")
End Sub
```

Example Output

```
12:00:01 AM
```

Trim$ — Function

Syntax `Trim[$](string$)`

Parameters

Name	Description
string$	Copy this string without the leading or trailing spaces.

Description Return the string with S$s leading and trailing spaces removed.

See Also `LTrim$( ), RTrim$( ).`

Example

```
Sub Main
    Debug.Print ".";Trim$("  x  ");"."
End Sub
```

Example Output

```
.x.
```

Type — Definition

Syntax

```
[Private|Public] Type name
  elem [(Dim[, …])] As type[...]
End Type
```

Description

Define a new *usertype*. Each *elem* defines an element of the type for storing data. As *type* defines the type of data that can be stored. A *User-defined type variable* has a value for each *elem*. Use *.elem* to access individual element values.

Public is assumed if neither *Private* or *Public* is specified.

Example

```
Type Employee
    Name As String
    Title As String
    Salary As Double
End Type

Sub Main
    Dim e As Employee
    e.Name = “John Doe”
    e.Title = “President”
    e.Salary = 100000
    Debug.Print e.Name   ‘"John Doe"
    Debug.Print e.Title  ‘"President"
    Debug.Print e.Salary ‘ 100000
End Sub
```

Example Output

```
John Doe
President
100000
```

TypeName — Function

Syntax

TypeName[$]**(***var***)**

Parameters

Name	Description
var	Return a string indicating the type of value stored in this variable.

Result

Value	Description
Empty	*Variant* variable is empty. It has never been assigned a value.
Null	*Variant* variable is null.

`Integer`	Variable contains an *integer* value.
`Long`	Variable contains a *long* value.
`Single`	Variable contains a *single* value.
`Double`	Variable contains a *double* value.
`Currency`	Variable contains a *currency* value.
`Date`	Variable contains a *date* value.
`String`	Variable contains a *string* value.
`Object`	Variable contains a *object* reference that is not Nothing. (An object may return a type name specific to that type of object.)
`Nothing`	Variable contains a *object* reference that is Nothing.
`Error`	Variable contains a error code value.
`Boolean`	Variable contains a *boolean* value.
`Variant`	Variable contains a variant value. (Only used for arrays of variants.)
`Unknown`	Variable contains a non-OLE Automation object reference.
`Byte`	Variable contains a byte value.
`( )`	Variable contains an array value. The TypeName of the element followed by ( ).

Description Return a string indicating the type of value stored in *var*.

See Also `VarType`.

Example

```
Sub Main
    Dim X As Variant
    Debug.Print TypeName(X)
    X = 1
    Debug.Print TypeName(X)
    X = 100000
    Debug.Print TypeName(X)
    X = 1.1
    Debug.Print TypeName(X)
    X = "A"
    Debug.Print TypeName(X)
    Set X = CreateObject("Word.Basic")
    Debug.Print TypeName(X)
    X = Empty
    X = Array(0,1,2)
    Debug.Print TypeName(X)
End Sub
```

Example Output

```
Empty
Integer
Long
Double
String
wordbasic
Variant()
```

UBound — Function

Syntax

```
UBound(var[, dimension])
```

Parameters

Name	Description
var	Return the highest index for this array variable.
dimension	Return the highest index for this dimension of *var*. If this is omitted then return the highest index for the first dimension.

Description

Return the highest index.

See Also

LBound( ).

Example

```
Sub Main
   Dim A(3,6)
   Debug.Print UBound(A)
   Debug.Print UBound(A,1)
   Debug.Print UBound(A,2)
End Sub
```

Example Output

```
 3
 3
 6
```

UCase$ — Function

Syntax

```
UCase[$](string$)
```

Parameters

Name	Description

string$	Return string value after all chars have been converted to uppercase.

Description Return a string from S$ where all the lowercase letters have been uppercased.

See Also LCase$( ).

Example

```
Sub Main
   Debug.Print UCase$("Hello")
End Sub
```

Example Output

```
HELLO
```

Unlock

Instruction

Syntax

Unlock *StreamNum*

-or-

Unlock *StreamNum, RecordNum*

-or-

Unlock *StreamNum, [start]* **To** *end*

Parameters

Name	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.
start	First record (or byte) in the range.
end	Last record (or byte) in the range.

Description Form 1: Unlock all of StreamNum.

Form 2: Unlock a record (or byte) of StreamNum.

Form 3: Unlock a range of records (or bytes) of StreamNum. If start is omitted then unlock starting at the first record (or byte).

Note: For sequential files (Input, Output and Append) unlock always affects the entire file.

See Also `Lock, Open.`

Example

```
Sub Main
    Dim V As Variant
    Open “SAVE_V.DAT” For Binary As #1
    Lock #1
    Get #1, 1, V
    V = “Hello”
    Put #1, 1, V
    Unlock #1
    Close #1
End Sub
```

Uses

Comment

Syntax ‘#Uses “module”

-or-

‘$Include: “module”

Description The Uses comment indicates that the current macro/module uses public symbols from the module.

See Also Class Module, Code Module, Object Module.

Example

```
‘Macro A.WWB
‘#Uses “B.WWB”
Sub Main
    Debug.Print BFunc$(“Hello”) ‘"HELLO"
End Sub

‘Module B.WWB
Public Function BFunc$(S$)
    BFunc$ = UCase(S$)
End Sub
```

Val — Function

Syntax `Val(string$)`

Parameters

Name	Description
string$	Return the number value for this string value. A string value beginning with &O is an octal number. A string value beginning with &H is a hex number. Otherwise it is decimal number.

Description Return the value of the *string$*.

Example

```
Sub Main
   Debug.Print Val("-1000")
End Sub
```

Example Output

```
-1000
```

VarType — Function

Syntax `VarType(var)`

Parameters

Name	Description
var	Return a number indicating the type of value stored in this variable.

Result

Value	Description
0	*Variant* variable is empty. It has never been assigned a value.
1	*Variant* variable is null.
2	Variable contains an *integer* value.
3	Variable contains a *long* value.
4	Variable contains a *single* value.
5	Variable contains a *double* value.
6	Variable contains a *currency* value.
7	Variable contains a *date* value.
8	Variable contains a *string* value.
9	Variable contains a *object* reference.

10	Variable contains a error code value.
11	Variable contains a *boolean* value.
12	Variable contains a variant value. (Only used for arrays of variants.)
13	Variable contains a non-OLE Automation object reference.
17	Variable contains a byte value.
+8192	Variable contains an array value. Use VarType( ) And 255 to get the type of element stored in the array.

Description Return a number indicating the type of value stored in *var*.

See Also `TypeName.`

Example

```
Sub Main
    Dim X As Variant
    Debug.Print VarType(X)
    X = 1
    Debug.Print VarType(X)
    X = 100000
    Debug.Print VarType(X)
    X = 1.1
    Debug.Print VarType(X)
    X = "A"
    Debug.Print VarType(X)
    Set X = CreateObject("Word.Basic")
    Debug.Print VarType(X)
    X = Empty
    X = Array(0,1,2)
    Debug.Print VarType(X)
End Sub
```

Example Output

```
 0
 2
 3
 5
 8
 9
 8204
```

VoltageRatioTodB — Function

Syntax `VoltageRatioTodB(num)`

Parameters

Name	Description
num	

Description Return the value in dB of the voltage ratio of *num* to 1.

Example

```
Sub Main
   Debug.Print Format(VoltageRatioTodB(2), "#.0000")
Sub
```

Example Output

```
 6.0206
```

Equation `VoltageRatio = 20 * Log10(num)`

Wait — Function

Syntax `Wait Delay`

Description Wait for *Delay* seconds.

Example

```
Sub Main
   Wait 5  'Wait for 5 seconds.
End Sub
```

WaitAndDoEvents — Instruction

Syntax `WaitAndDoEvents Delay`

Description Wait for Delay seconds while giving other events on the computer time to continue. This is the preferred over Wait if any other activity needs to be kept running efficiently (such as APWIN sweeps). Because other events are kept running, timing will be slightly less accurate than if Wait is used.

See Also Wait.

Example

```
Sub Main
   WaitAndDoEvents 5 ' wait for 5 seconds
End Sub
```

Weekday — Function

Syntax `Weekday(dateexpr)`

Parameters

Name	Description
dateexpr	Return the weekday for this date value.

Description Return the weekday (1 to 7). Sunday=1, Monday=2, Tuesday=3, Wednesday=4, Thursday=5, Friday=6 and Saturday=7.

See Also Date( ), Day( ), Month( ), Year( ).

Example

```
Sub Main
    Debug.Print Weekday(#1/1/1996#)
End Sub
```

Example Output

```
2
```

While — Statement

Syntax

```
While condexpr
  statements
Wend
```

Description Execute *statements* while *condexpr* is **True**.

See Also Do, For, For Each, Exit While.

Example

```
Sub Main
    I = 2
    While I < 10
        I = I*2
    Wend
    Debug.Print I
End Sub
```

Example Output 16

With — Statement

Syntax

```
With objexpr
  statements
End With
```

Description *Method* and *property* references may be abbreviated inside a With block. Use *.method* or *.property* to access the object specified by the With *objexpr*.

Example

```
Sub Main
   Dim Excel As Object
   Set Excel = CreateObject("Excel.Application")
   With Excel
      Excel.Visible = True
      Excel.Quit
   End With
   Set Excel = Nothing
End Sub
```

WithEvents — Definition

Syntax

```
[Dim | Private | Public] _
WithEvents name As objtype[, ...]
```

Description Dimensioning a module level variable WithEvents allows the macro to implement event handling Subs. The variable's As type must be a type from a referenced type library (or language extension) which implements events.

Remarks This keyword is supported by the single DLL IDE/interpreter (aka the Enterprise edition). It is not supported by the interpreter implemented in WW_CU516.DLL or WW_CU532.DLL.

See Also `Dim, Private, Public.`

Example

```
Dim WithEvents X As Thing
Sub Main
    Set X = New Thing
    X.DoIt ‘ DoIt method raises DoingIt event
End Sub

Private Sub X_DoingIt
    Debug.Print “X.DoingIt event”
End Sub
```

Write — Instruction

Syntax **Write** *#streamnum, expr*[, ...]

Description Writes *expr*(s) to *Streamnum*. String values are quoted. Null values are written as #NULL#. Boolean values are written as #FALSE# or #TRUE#. Date values are written as #date#. Error codes are written as #Error number#.

See Also `Input, Line Input, Print.`

Example

```
Sub Main
    A = 1
    B = 2
    C$ = “Hello”
    Open “FILENAME.EXT” For Output As #1
    Write #1,A,B,C$
    Close #1
End Sub
```

Year — Function

Syntax **Year(***dateexpr***)**

Parameters

Name	Description
dateexpr	Return the year for this date value.

Description Return the year.

See Also `Date( ), Day( ), Month( ), Weekday( ).`

Example

```
Sub Main
   Debug.Print Year(#1/1/1996#)
End Sub
```

Example Output

```
1996
```

User Notes

User Notes

User Notes

User Notes

User Notes

Appendix A Terms

arglist [|*expr*|*param*:=*expr*][, ...]

A list of zero or more *exprs* that are assigned to the parameters of the sub, function or property.

A positional parameter may be skipped by omitting the expression. Only optional parameters may be skipped.

Positional parameter assignment is done with *expr*. Each parameter is assigned in turn. By name parameter assignment may follow.

By name parameter assignment is done with *param*:=*expr*. All following parameters must be assigned by name.

As [New] type Dim, Private, Public and Static statements may declare variable types using As type or As New objtype. A variable declared using As New objtype is automatically created prior to use, if the variable is Nothing.

As type Variable and argument types, as well as, function and property results may be specified using As type: *Boolean*, *Byte, Currency*, *Date*, Double, *Integer*, *Long*, *Object, Single, String, String*n, UserDialog, Variant, usertype*.

attribute A file attribute is zero or more of the following values added together.

Value	Description
0	Normal file.
1	Read-only file.
2	Hidden file.
4	System file.
8	Volume label.
16	MS-DOS directory.
32	File has changes since last backup.

big-endian Multiple byte data values (not strings) are stored with the highest order byte first. For example, the long integer &H01020304 is stored as this sequence of four bytes: &H01, &H02, &H03 and &H04. A Binary or Random file written using Put uses little-endian format so that it can be read using Get on any machine. (Big-endian machines, like the

Power-PC, reverse the bytes as they are read by Get or written by Put.)

See Also: Dir( ), GetAttr( ), SetAttr( ).

charlist A group of one or more characters enclosed by [ ] as part of Like operator's right string expression.

o This list contains single characters and/or character ranges which describe the characters in the list.

o A range of characters is indicated with a hyphen (-) between two characters. The first character must be ordinally less than or equal to the second character.

o Special pattern characters like ?, *, # and [can be matched as literal characters.

o The] character can not be part of charlist, but it can be part of the pattern outside the charlist.

condexpr An expression that returns a numeric result. If the result is zero then the conditional is False. If the result is non-zero then the conditional is True.

```
0 false
-1 true
X > 20 true if X is greater than 20
S$ = hello true if S$ equals hello
```

dateexpr An expression that returns a *date* result. Use #literal-date# to express a date value.

```
#1/1/2000#  Jan 1, 2000
Now+7  seven days from now
DateSerial(Year(Now)+1,Month(Now),Day(Now))  one year
from now
```

dialogfunc A dialog function executes while a *UserDialog* is visible.

dim [lower To] upper

Array dimension. If lower is omitted then the lower bound is zero. upper must be at least as big as lower.

Dim A(100 To 200) '101 values

Note: For ReDim the lower and upper may be any valid expression. Otherwise, lower and upper must be constant expressions.

dlgvar A dialog variable holds values for fields in the dialog. Dialog variables are declared using **Dim** dlgvar As *UserDialog*.

expr An expression that returns the appropriate result.

field Use .field to access individual fields in a dialog variable.

```
dlg.Name$
dlg.ZipCode
```

instruction A single command.

```
Beep
Debug.Print Hello
Today = Date
```

Multiple instructions may be used instead of a single instruction by separating the single instructions with colons.

```
X = 1:Debug.Print X
If X = 1 Then Debug.Print X=;X:Stop
Beep  must resume from Stop to get to here
```

label An identifier that *names* a statement. Identifiers start with a letter. Following chars may be a letter, an underscore or a digit.

little-endian Multiple byte data values (not strings) are stored with the lowest order byte first. For example, the long integer &H01020304 is stored as this sequence of four bytes: &H04, &H03, &H02 and &H01. A Binary or Random file written using Put uses little-endian format so that it can be read using Get on any machine. (Big-endian machines, like the Power-PC, reverse the bytes as they are read by Get or written by Put.)

macro A macro is like an application. Execution starts at the macro's Sub Main.

method An object provides methods and *properties*. Methods can be called as subs (the return value is ignored), or used as functions (the return value is used).

If the method name contains characters that are not legal in a *name*, surround the method name with [].

```
App.[Title$]
```

module A file with public symbols that are accessible by other modules/macros via the #Uses comment.

o A module is loaded on demand.

o A code module is a code library.

o An object module or class module implements an OLE automation object.

o A module may also access other modules with its own #Uses comments.

name An identifier that names a variable or a user defined subroutine, function or property. Identifiers start with a letter. Following chars may be a letter, an underscore or a digit.

```
Count
DaysTill2000
Get_Data
```

num An expression that returns a numeric result. Use &O to express an octal number. Use &H to express a hex number.

```
10236
3.14159
1.2E12
Count
Count-1
InStr(S$,"A")
&O100 64
&H100 256
```

numvar A variable that holds one numeric value. The name of a numeric variable may be followed by the appropriate *type* char.

objexpr A expression that returns a reference to an object.

```
CreateObject(WinWrap.CDemoApplication)
```

objtype A specific OLE type defined by your application, another application or by an object module or class module.

See Also: Objects, CreateObject( ), GetObject( ).

objvar A variable that holds a *objexpr* which references an object. Object variables are declared using As *Object* in a **Dim**, **Private** or **Public** statement.

param [[Optional] [| ByVal | ByRef] | ParamArray] param[type][( )] [As type]

The *param* receives the value of the associated expression in the subroutine, function or property call. (See *arglist*.)

An Optional *param* may be omitted from the call. It must be a **Variant** type. All parameters following an Optional parameter must also be Optional.

ParamArray may be used on the final *param*. It must be an array of **Variant** type. It must not follow any Optional parameters. The ParamArray receives all the expressions at the end of the call as an array. If **LBound**(*param*) **UBound**(*param*) then the ParamArray didnt receive any expressions.

If the *param* is not ByVal and the expression is merely a variable then the *param* is a reference to that variable (ByRef). (Changing *param* changes the variable.) Otherwise, the parameter variable is local to the subroutine, function or property, so changing its value does not affect the caller.

Use *param*( ) to specify an array parameter. An array parameter must be referenced and can not be passed by value. The bounds of the parameter array are available via **LBound( )** and **UBound( )**.

Property Get, Let and Set blocks do not allow Optional or ParamArray parameter types.

precedence When several operators are used in an expression, each operator is evaluated in a predetermined order. Operators are evaluated in this order:

^ (power)

- (negate)

* (multiply), / (divide)

\ (integer divide)

Mod (integer remainder)

+ (add), - (difference)

& (string concatenate)

= (equal), <> (not equal), < (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to), **Is** (object equivalence)

Not (logical bitwise invert)

And (logical bitwise and)

Or (logical or bitwise or)

Xor (logical or bitwise exclusive-or)

Eqv (logical or bitwise equivalence)

Imp (logical or bitwise implication)

Operators shown on the same line are evaluated from left to right.

property An object provides *methods* and properties. Properties may be used as values (like a function call) or changed (using assignment syntax).

If the property name contains characters that are not legal in a *name*, surround the property name with [].

```
App.[Title$]
```

statement One or more *instructions*. A statement is at least one macro line long. **Begin** Dialog, **Do**, **For**, **If** (multiline), **Select Case**, **While** and **With** statements are always more than one line long. A single line statement continues on the next line if it ends a line with a space and an underscore _.

```
S$ = This long string is easier to read,  + _
if it is broken across two lines.
Debug.Print S$
```

str An expression that returns a string result.

```
Hello
S$
S$ +  GoodbyeS$ &  Goodbye
Mid$(S$,2)
```

strarray A variable that holds an array of string values. The name of a string variable may be followed by a $.

strvar A variable that holds one string value. The name of a string variable may be followed by a $.

```
FirstName$
```

type Variable and argument types, as well as, function and property results may be specified using a type character as the last character in their name.

Type char	As Type
%	Integer
&	Long
!	Single
#	Double
@	Currency
$	String

userenum User defined enums are defined with Enum.

usertype User-defined types are defined with **Type**.

usertypevar A user-defined type variable holds values for elements of the user-defined type. Use r-defined types are defined using **Type**. User-defined variables are declared using **Dim**, **Private** or **Public**.

var A variable holds either a string, a numeric value or an array of values depending on its type.

variantvar A variant variable holds any type of value (except *String*n* or *usertypevar*).

Appendix B Error List

The following table lists all error codes with the associated error text.

Error #	Description
10000	Macro execution interrupted.
10001	Out of memory.
10008	Invalid '#Uses "module" comment.
10009	Invalid '#Uses module dependency.
10010	Macro is already running.
10011	Cant allocate memory to macro.
10012	Macro has syntax errors.
10013	Macro does not exist.
10014	Another macro is paused and cant continue at this time.
10017	No macro is currently active.
10018	Subroutine does not exist.
10019	Wrong number of parameters.
10021	Cant allocate large array.
10022	Array is not dimensioned.
10023	Array index out of range.
10024	Array lower bound is larger than upper bound.
10025	Array has a different number of indexes.
10030	User dialog has not been defined.
10031	User pressed cancel.
10032	User dialog item id is out of range.
10033	No UserDialog is currently displayed.
10034	Current UserDialog is inaccessible.
10035	Wrong with, dont GoTo into or out of With blocks.
10040	Module could not be loaded.
10041	Function not found in module.
10048	File not opened with read access.
10049	File not opened with write access.
10050	Record length exceeded.
10051	Could not open file.
10052	File is not open.
10053	Attempt to read past end-of-file.

10054	Expecting a stream number 1, 2, 3 or 4.
10055	Input does not match var type.
10056	Expecting a length in the range 1 to 32767.
10057	Stream number is already open.
10058	File opened in the wrong mode for this operation.
10059	Error occurred during file operation.
10060	Expression has an invalid floating point operation.
10061	Divide by zero.
10062	Overflow.
10063	Expression underflowed minimum representation.
10064	Expression loss of precision in representation.
10069	String value is not a valid number.
10071	Resume can only be used in an On Error handler.
10075	Null value cant be used here.
10080	Type mismatch.
10081	Type mismatch for parameter #1.
10082	Type mismatch for parameter #2.
10083	Type mismatch for parameter #3.
10084	Type mismatch for parameter #4.
10085	Type mismatch for parameter #5.
10086	Type mismatch for parameter #6.
10087	Type mismatch for parameter #7.
10088	Type mismatch for parameter #8.
10089	Type mismatch for parameter #9.
10090	OLE Automation error.
10091	OLE Automation: no such property or method.
10092	OLE Automation: server cannot create object.
10093	OLE Automation: server cannot load file.
10094	OLE Automation: Object var is Nothing.
10095	OLE Automation: server could not be found.
10096	OLE Automation: no object currently active.
10097	OLE Automation: wrong number of parameters.
10098	OLE Automation: bad index.
10099	OLE Automation: no such named parameter.
10100	Directory could not be found.

10101	File could not be killed.
10102	Directory could not be created.
10103	File could not be renamed.
10104	Directory could not be removed.
10105	Drive not found.
10106	Source file could not be opened.
10107	Destination file could not be created.
10108	Source file could not be completely read.
10109	Destination file could not be completely written.
10110	Missing close brace }.
10111	Invalid key name.
10112	Missing close paren).
10113	Missing close bracket].
10114	Missing comma ,.
10115	Missing semi-colon ;.
10116	SendKeys couldnt install the Windows journal playback hook.
10119	String too long (too many keys).
10120	Window could not be found.
10130	DDE is not available.
10131	Too many simultaneous DDE conversations.
10132	Invalid channel number.
10133	DDE operation did not complete in time.
10134	DDE server died.
10135	DDE operation failed.
10140	Cant access the clipboard.
10150	Window style must be in the range from 1 to 9.
10151	Shell failed.
10160	Declare is not implemented.
10200	Basic is halted due to an unrecoverable error condition.
10201	Basic is busy and can't provide the requested service.
10202	Basic call failed.
10203	Handler property: prototype specification is invalid.
10204	Handler is already in use.

APWIN BASIC
User's Manual and Programmer's Reference

Volume 1 Language Reference

Volume 2 Extensions A-R

Volume 3 Extensions S-Z

Version 1.52
November, 1998

Version 1.52 november, 1998

Published by:

Audio precision

Audio Precision, Inc.
PO Box 2209
Beaverton, Oregon 97075-2209
U.S. Toll Free: 1-800-231-7350
Tel: (503) 627-0832 Fax: (503) 641-8906
email: techsupport@audioprecision.com
Web: www.audioprecision.com

Printed in the United States of America
Audio Precision Part # 8211-0012

APWIN Basic Extensions Reference

Introduction

This chapter of the manual is divided into three sections.

The first section consists of APWIN 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 APBASIC extensions.

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

Part	Description
Syntax:	Programming usage information.
Command type:	Method or Property
Data type:	Setting Data Type.
Result:	Query Data Type.
Description:	Technical Information.
See also:	Commands related to the current command that may contain relevant information.
Example:	Example procedure/macro
Example Output:	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.
Comments:	Additional information relating to the example procedure/macro.

Manual Conventions

This manual uses the following typographic conventions.

Example	Description
event, *var, arg*	For the syntax part of each command, italicized words indicate placeholders where the user must enter additional information.
FILENAME.TXT	Words in all CAPITOL letters indicate file names.

`Sub Main`
`  AP.Gen.Amp = 1.0`
`End Sub`
	This font is used in all example macros and code modules.

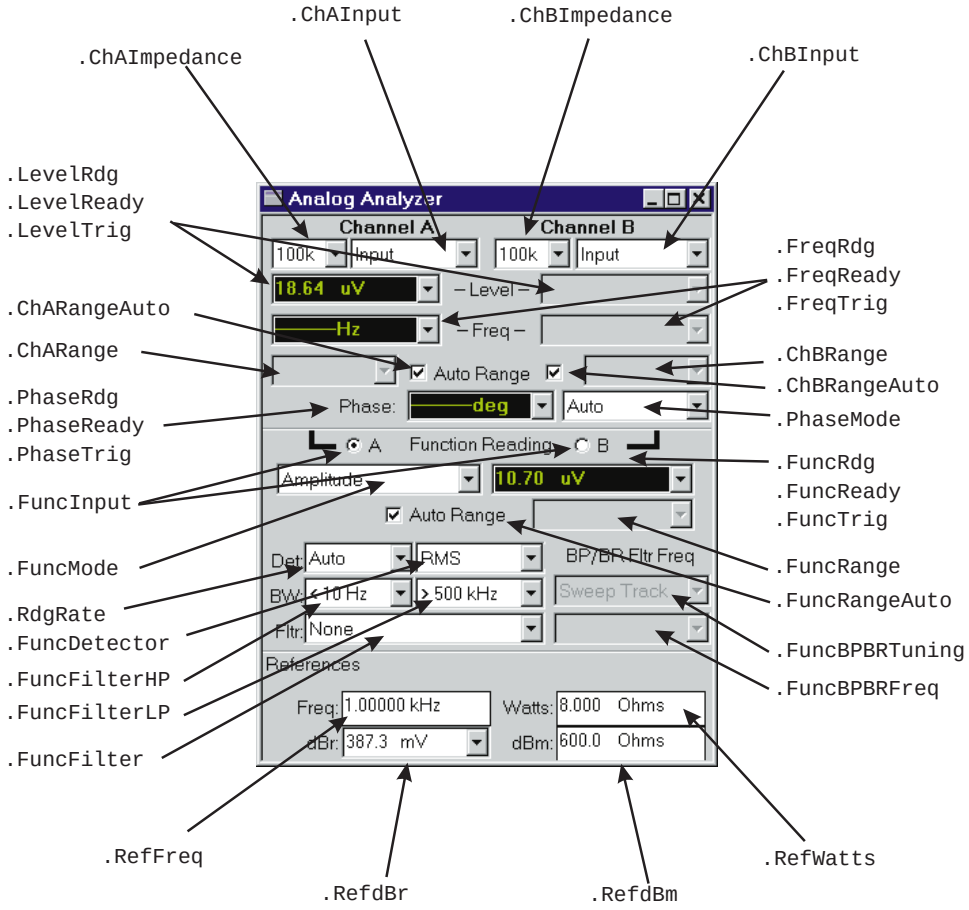
`[`*`expression list`*`]`	In syntax, items inside square brackets are optional.
`{`*`While`*` | `*`Until`*`}`	In syntax, braces and a vertical bar indicate a choice between two or more items.
Command	For the syntax part of each command, the bold characters identify the part of the command that must be entered.
`AP.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.
❶	This symbol denotes a command or example procedure that is applicable to System One.
❷	This symbol denotes a command or example procedure that is applicable to System Two.

System Panels Listed Alphabetically by Title

System One Panels

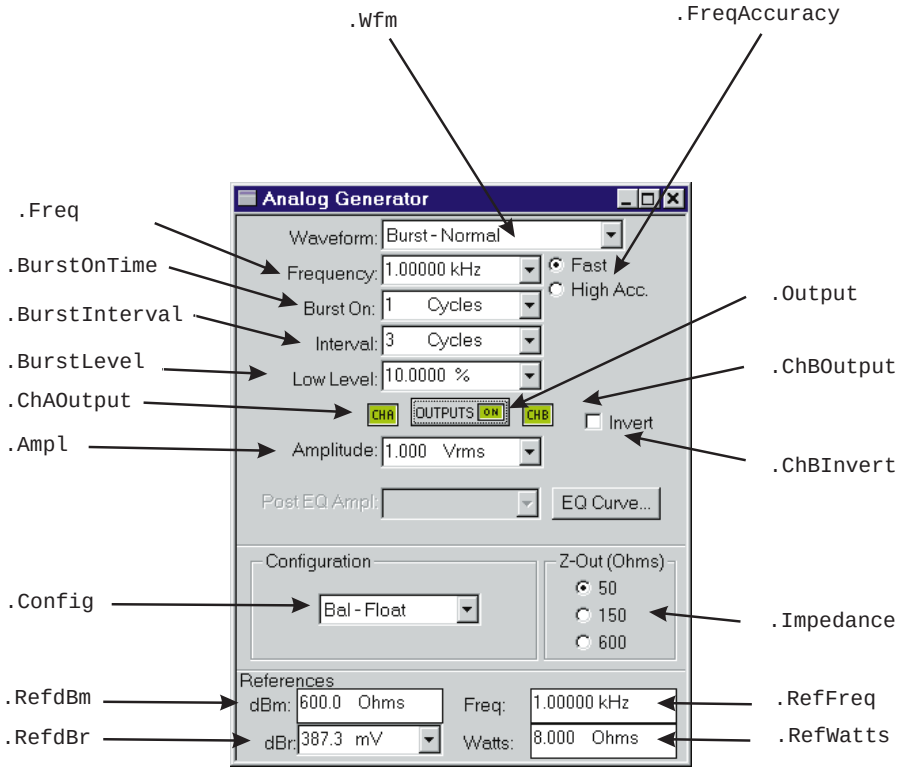
❶ Analog Analyzer

All commands on this page start with the following:
AP.Anlr
Example: AP.Anlr.ChAImpedance



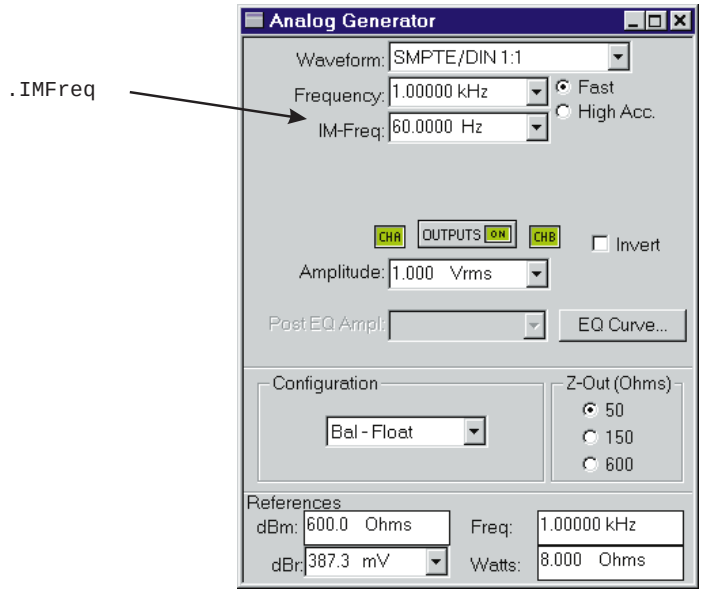
❶ Analog Generator ...

All commands on this page start with the following:
AP.Gen
Example: AP.Gen.Freq



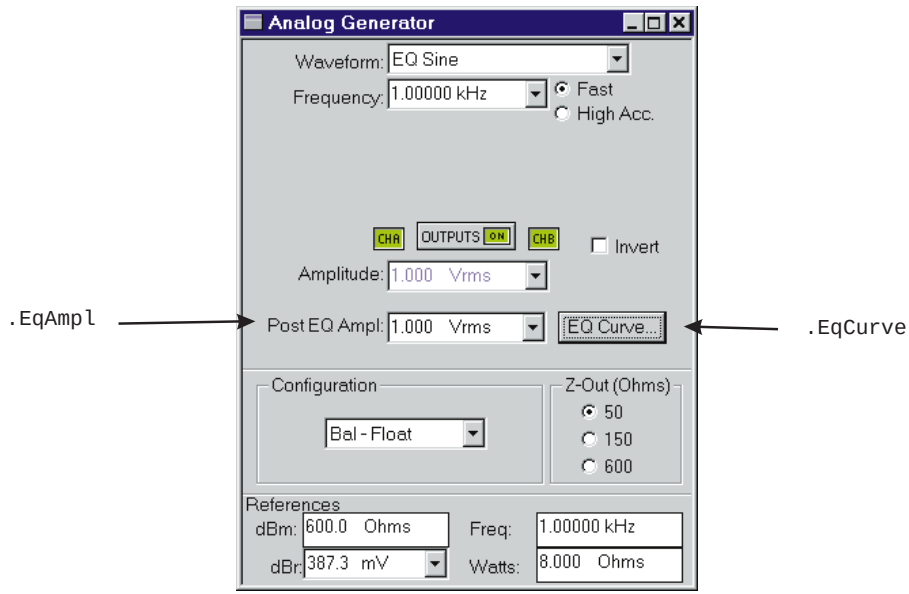
❶ Analog Generator Continued ...

All commands on this page start with the following:
AP.Gen
Example: AP.Gen.IMFreq



❶ Analog Generator Continued

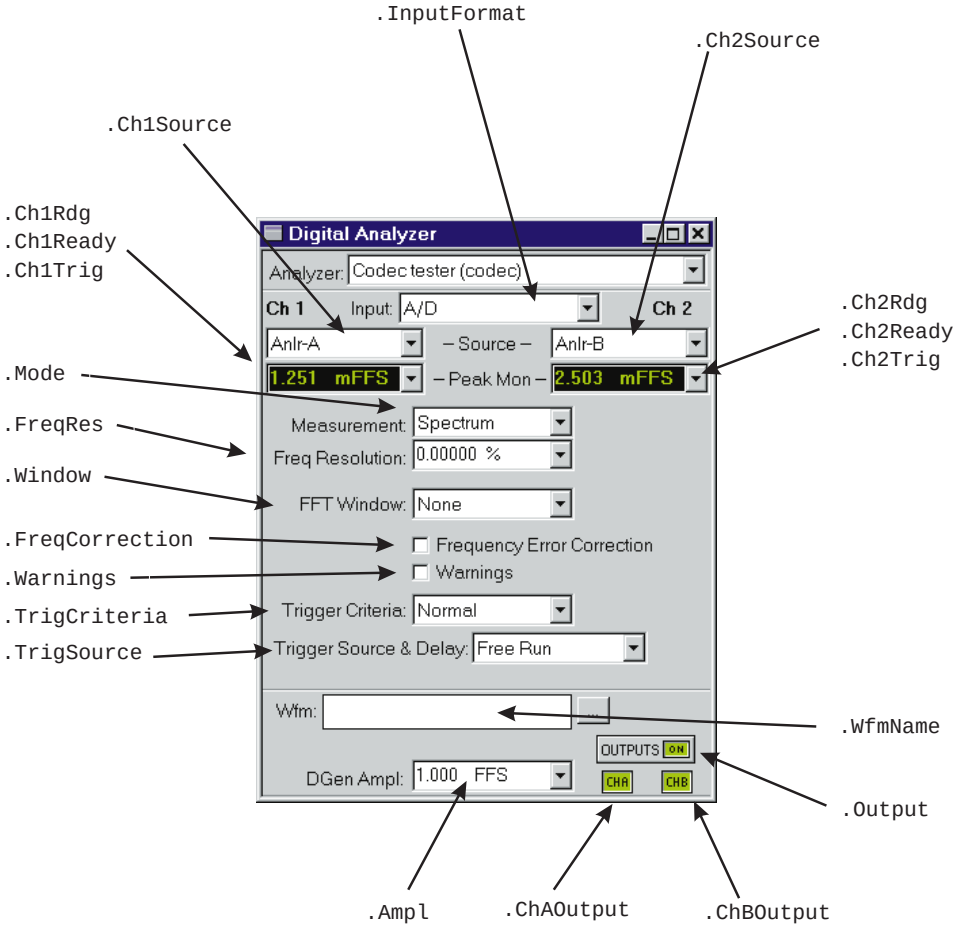
All commands on this page start with the following:
AP.Gen
Example: AP.Gen.Freq



Digital Analyzer Panels

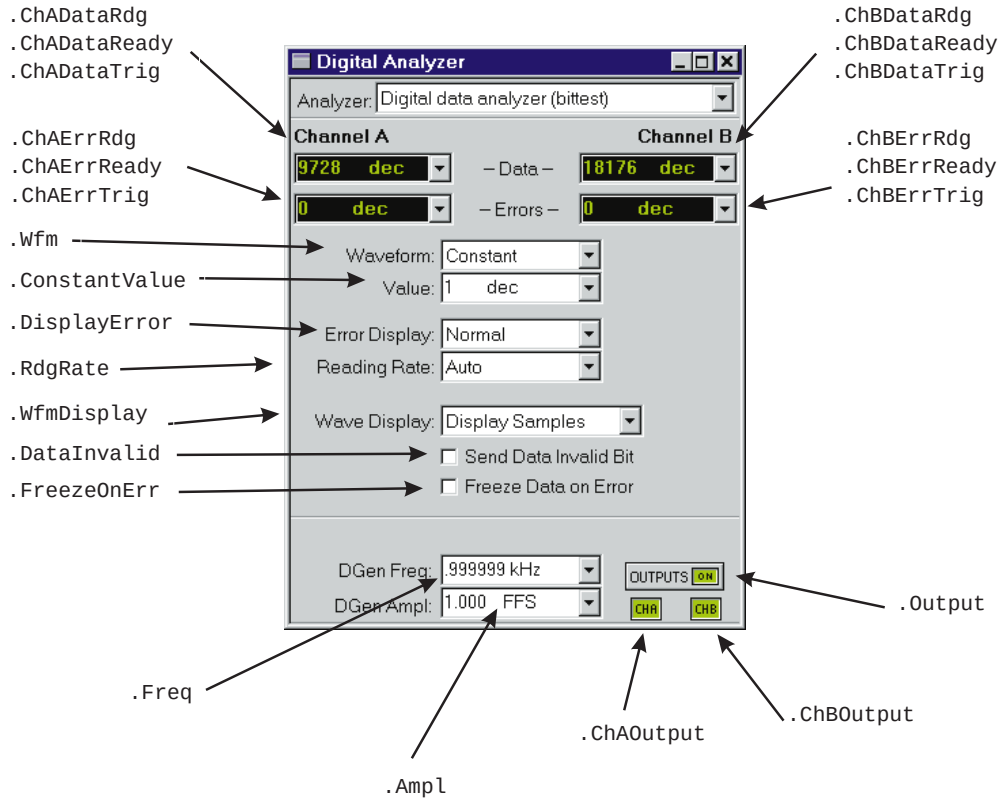
❶ Codec Tester (CODEC)

All commands on this page start with the following:
AP.S1DSP.Codec
Example: AP.S1DSP.Codec.InputFormat



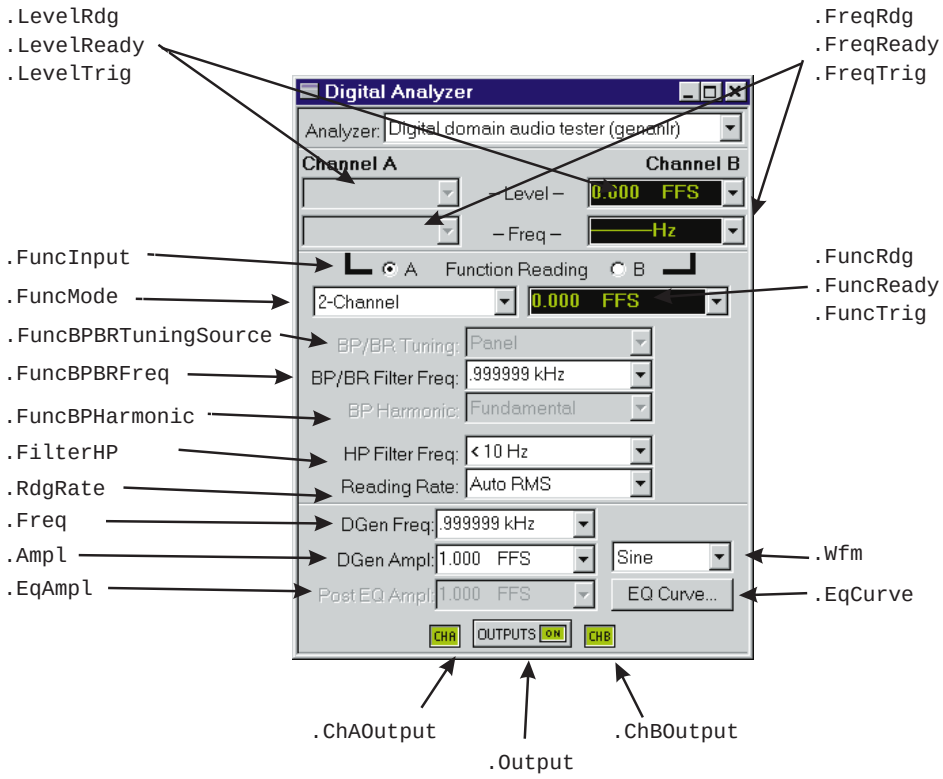
❶ Digital Data Analyzer (BITTEST)

All commands on this page start with the following:
AP.S1DSP.Bittest
Example: AP.S1DSP.Bittest.ChADataRdg



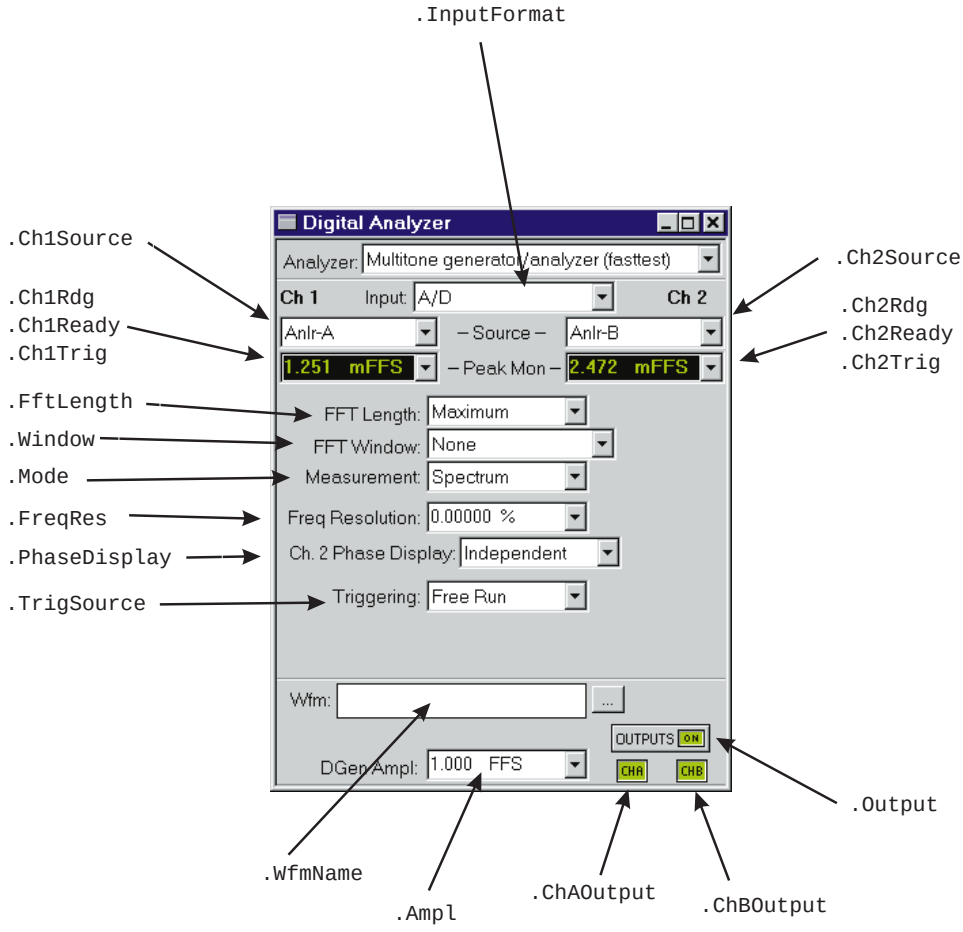
❶ Digital Domain Audio Tester (GENANLR)

All commands on this page start with the following:
AP.S1DSP.GenAnlr
Example: AP.S1DSP.GenAnlr.ChALevelRdg



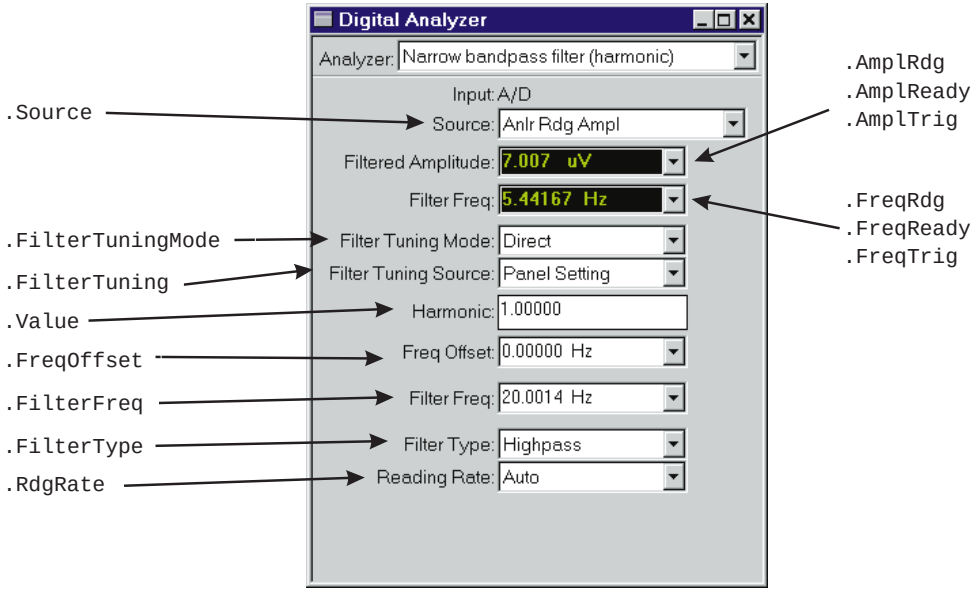
❶ Multitone Generator/Analyzer (FASTTEST)

All commands on this page start with the following:
AP.S1DSP.Fasttest
Example: AP.S1DSP.FastTest.InputFormat



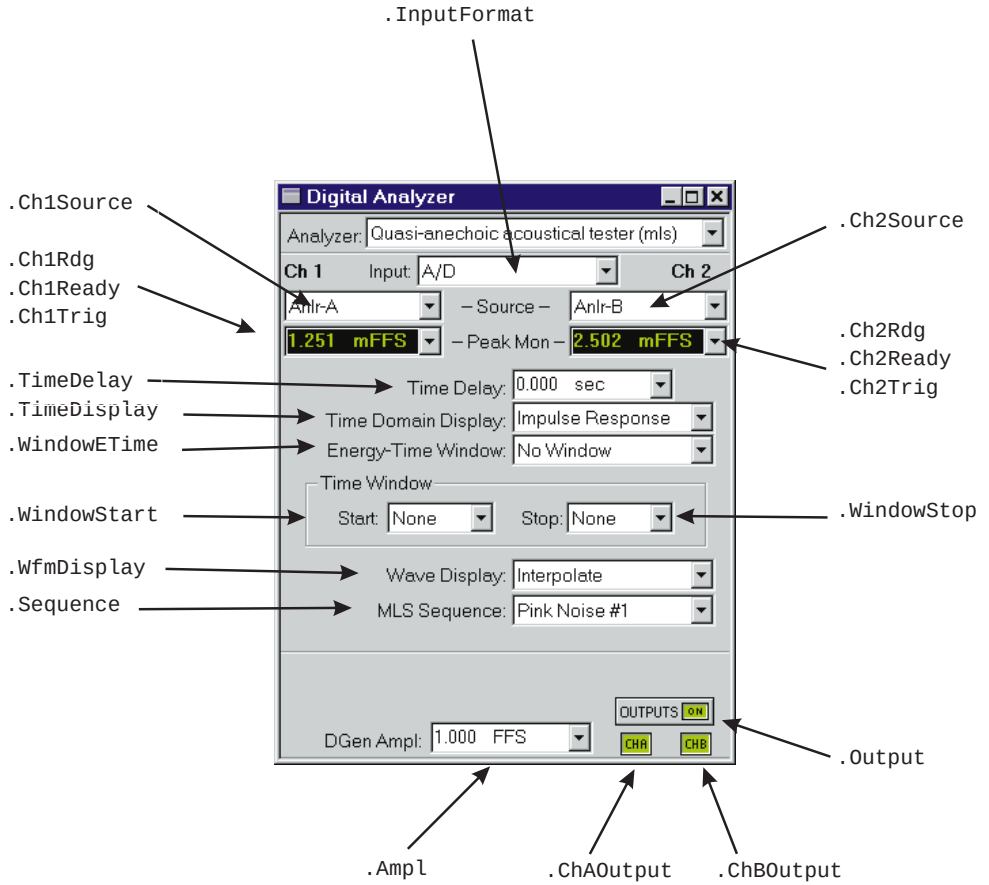
❶ Narrow Bandpass Filter (HARMONIC)

All commands on this page start with the following:
AP.S1DSP.Harmonic
Example: AP.S1DSP.Harmonic.Value



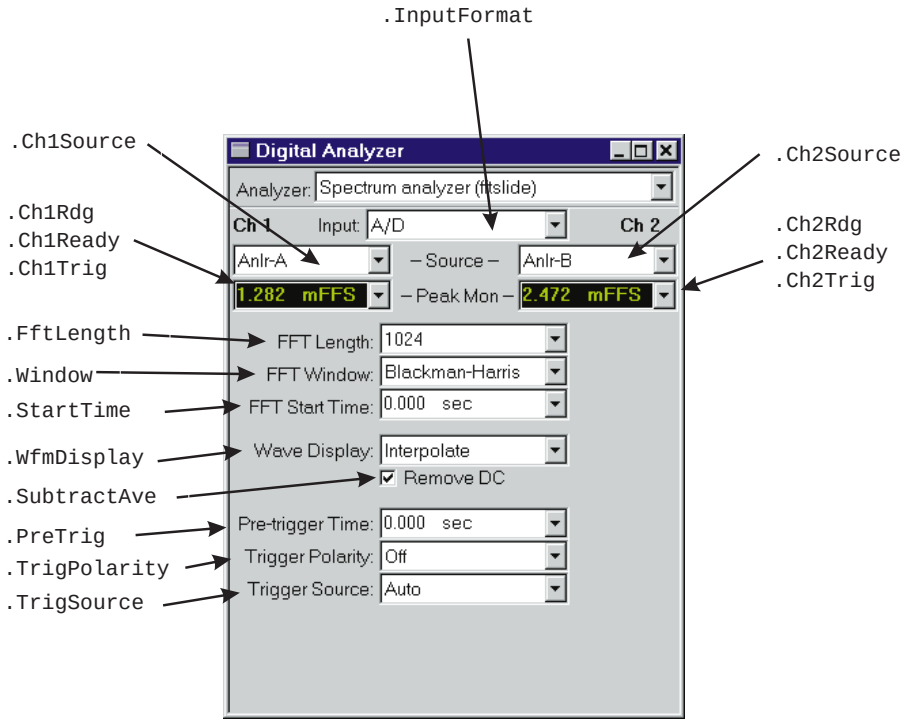
❶ Quasi-Anechoic Acoustical Tester (MLS)

All commands on this page start with the following:
AP.S1DSP.Mls
Example: AP.S1DSP.Mls.InputFormat



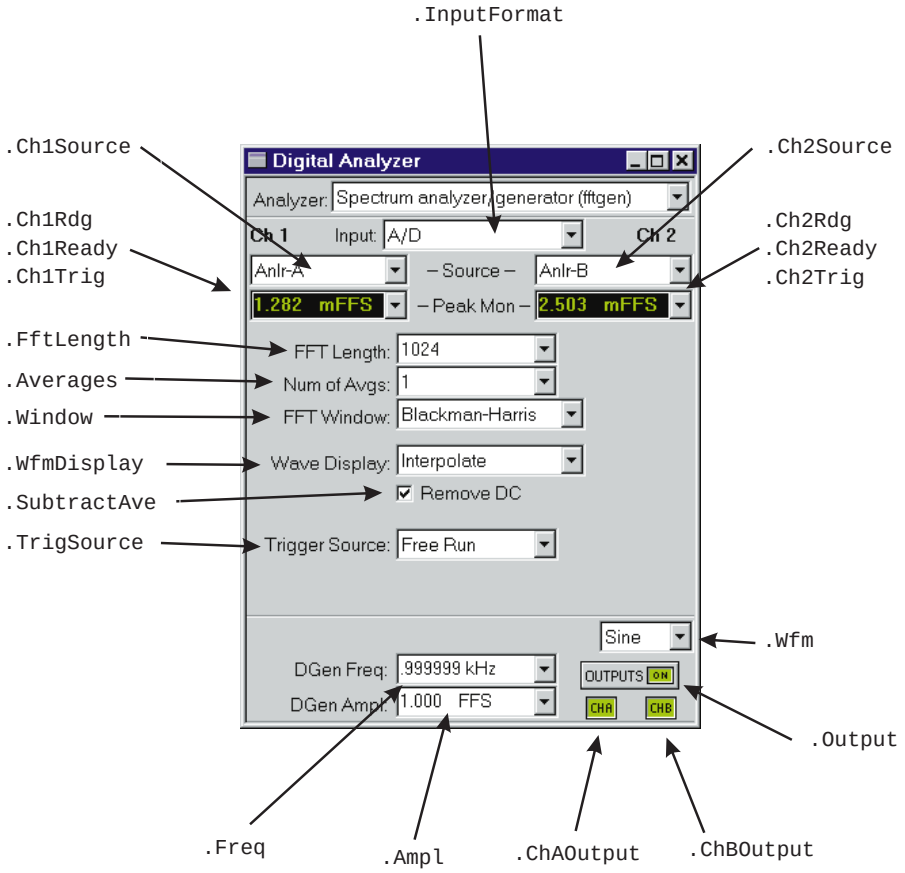
❶ Spectrum Analyzer (FFTSLIDE)

All commands on this page start with the following:
AP.S1DSP.FFTSlide
Example: AP.S1DSP.FFTSlide.InputFormat



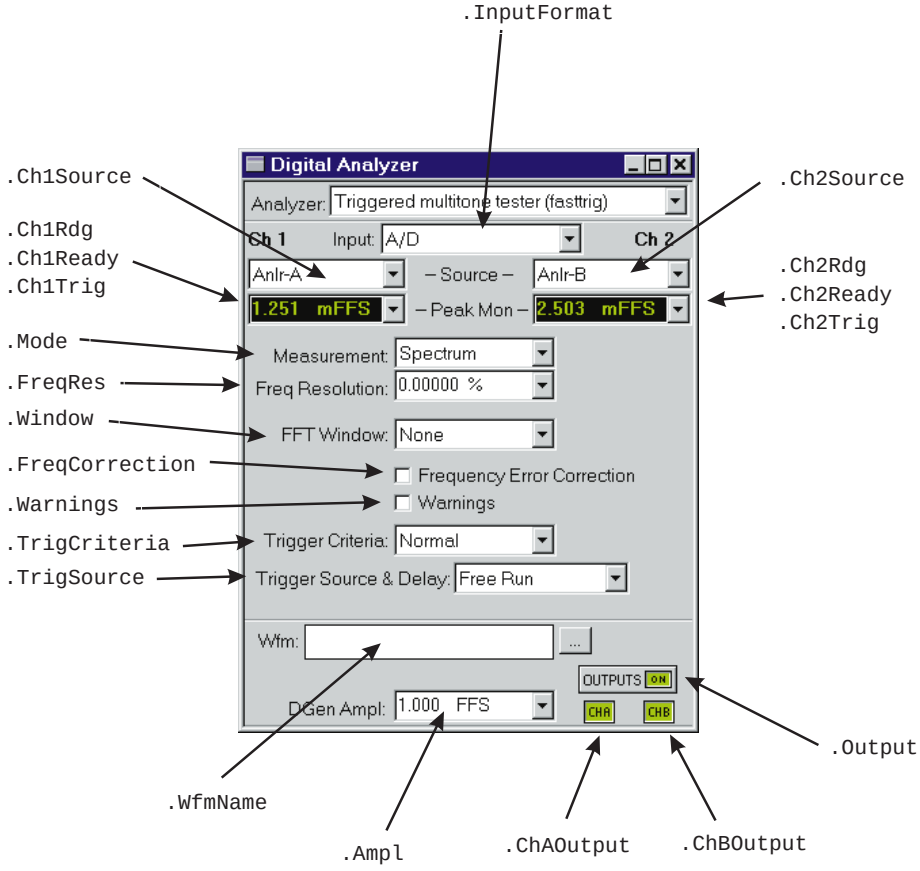
❶ Spectrum Analyzer/Generator (FFTGEN)

All commands on this page start with the following:
AP.S1DSP.FFTGen
Example: AP.S1DSP.FFTGen.InputFormat



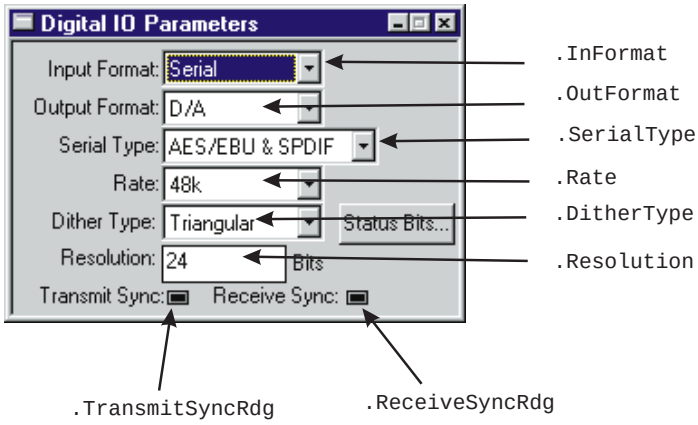
❶ Triggered Multitone Tester (FASTTRIG)

All commands on this page start with the following:
AP.S1DSP.FastTrig
Example: AP.S1DSP.FastTrig.InputFormat

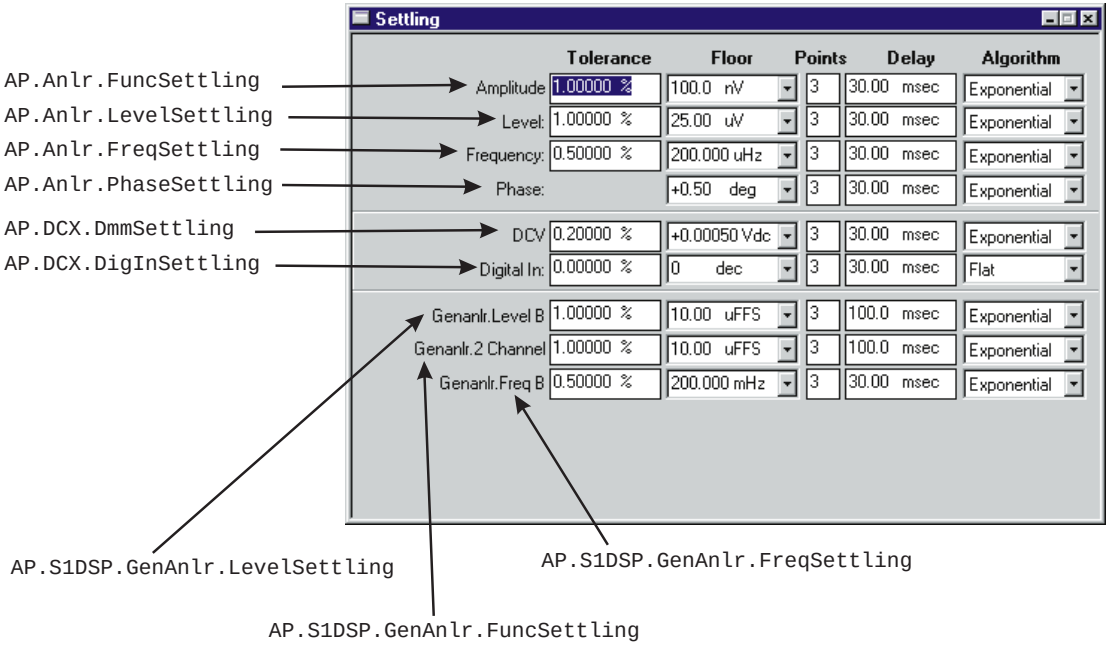


❶ Digital IO Parameters

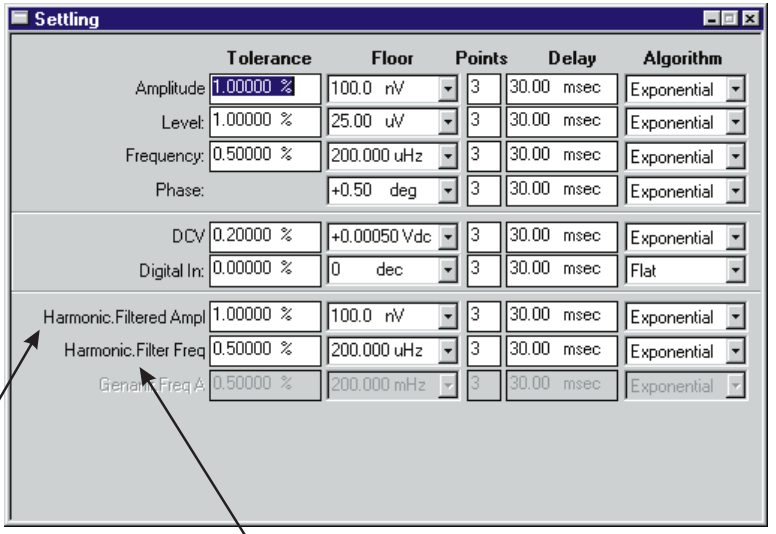
All commands on this page start with the following:
AP.S1Dio
Example: AP.S1Dio.InFormat



❶ Settling ...

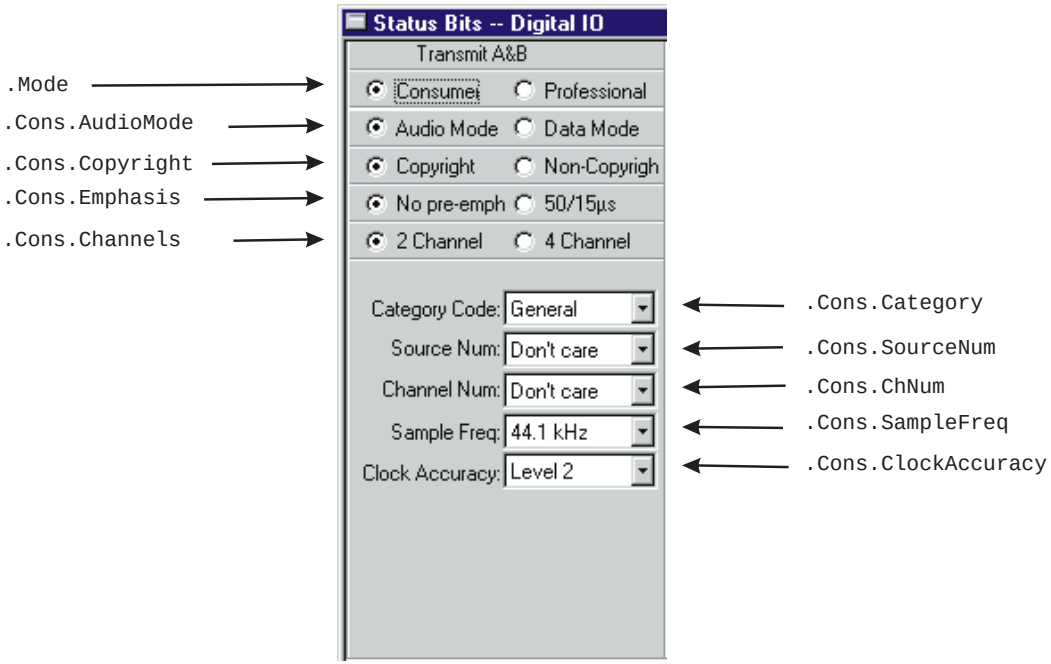

AP.Anlr.FuncSettling
AP.Anlr.LevelSettling
AP.Anlr.FreqSettling
AP.Anlr.PhaseSettling
AP.DCX.DmmSettling
AP.DCX.DigInSettling
Settling
Tolerance
Floor
Points
Delay
Algorithm
Amplitude 1.00000 % 100.0 nV 3 30.00 msec Exponential
Level: 1.00000 % 25.00 uV 3 30.00 msec Exponential
Frequency: 0.50000 % 200.000 uHz 3 30.00 msec Exponential
Phase: +0.50 deg 3 30.00 msec Exponential
DCV 0.20000 % +0.00050 Vdc 3 30.00 msec Exponential
Digital In: 0.00000 % 0 dec 3 30.00 msec Flat
Genanlr.Level B 1.00000 % 10.00 uFFS 3 100.0 msec Exponential
Genanlr.2 Channel 1.00000 % 10.00 uFFS 3 100.0 msec Exponential
Genanlr.Freq B 0.50000 % 200.000 mHz 3 30.00 msec Exponential
AP.S1DSP.GenAnlr.LevelSettling
AP.S1DSP.GenAnlr.FuncSettling
AP.S1DSP.GenAnlr.FreqSettling

❶ Settling Continued



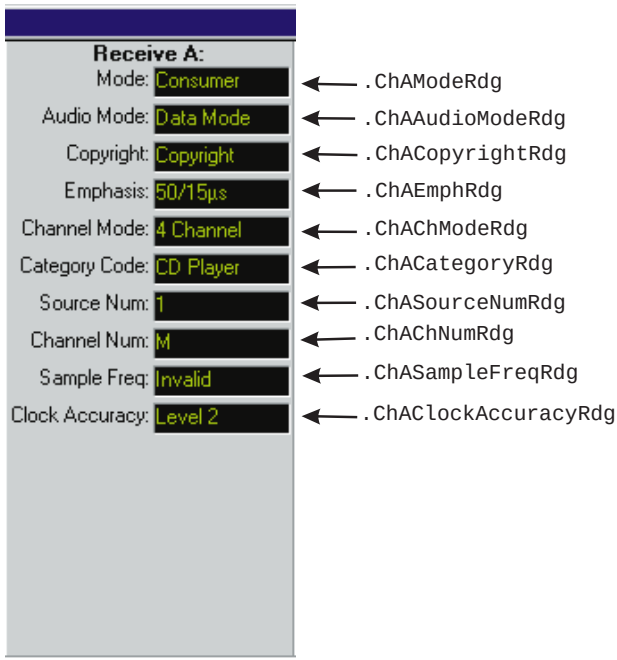
❶ Status Bits — Digital IO - Transmit Consumer

All commands on this page start with the following:
AP.Bits
Example: AP.Bits.Mode



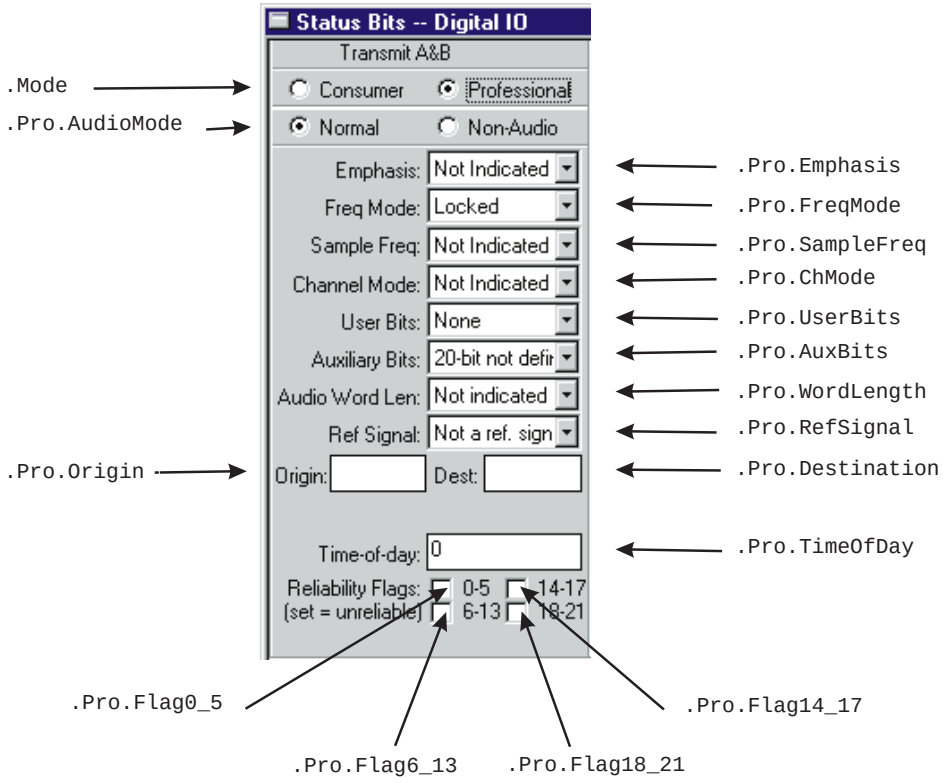
❶ Status Bits — Digital IO - Receive Consumer / Professional

All commands on this page start with the following:
AP.Bits
Example: AP.Bits.ChABitsStatusXferToArray



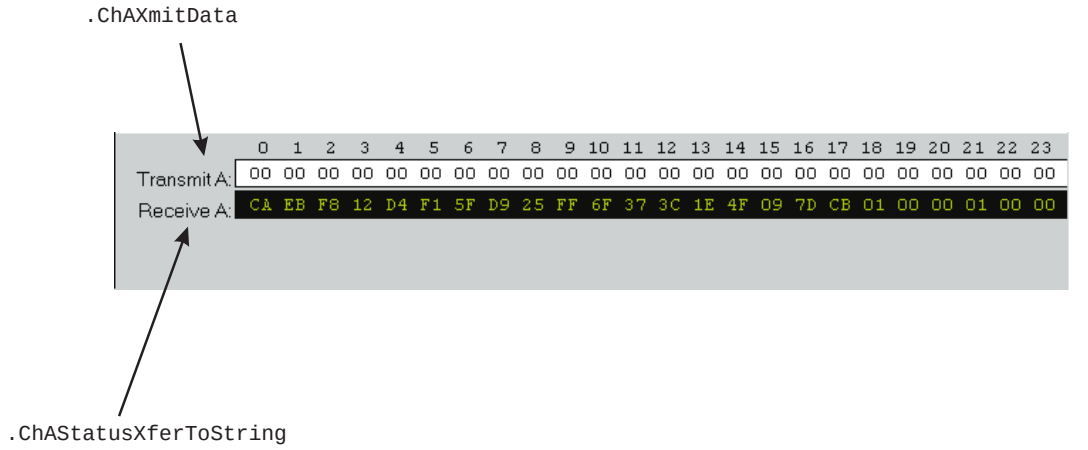
❶ Status Bits — Digital IO - Transmit Professional

All commands on this page start with the following:
AP.Bits
Example: AP.Bits.Mode



❶ Status Bits — Digital IO - Transmit/Receive

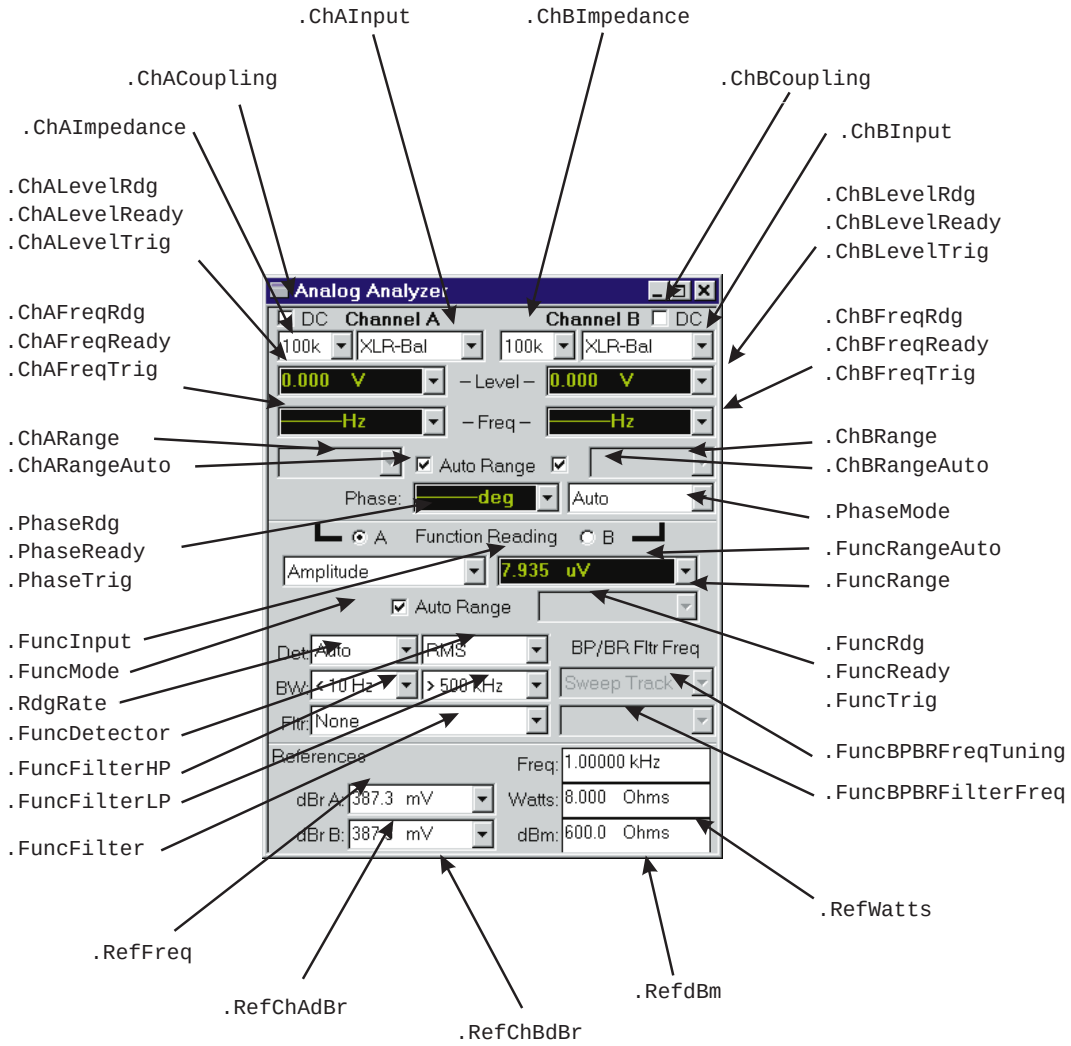
All commands on this page start with the following:
AP.Bits
Example: AP.Bits.ChABitsStatusXferToArray



System Two Panels

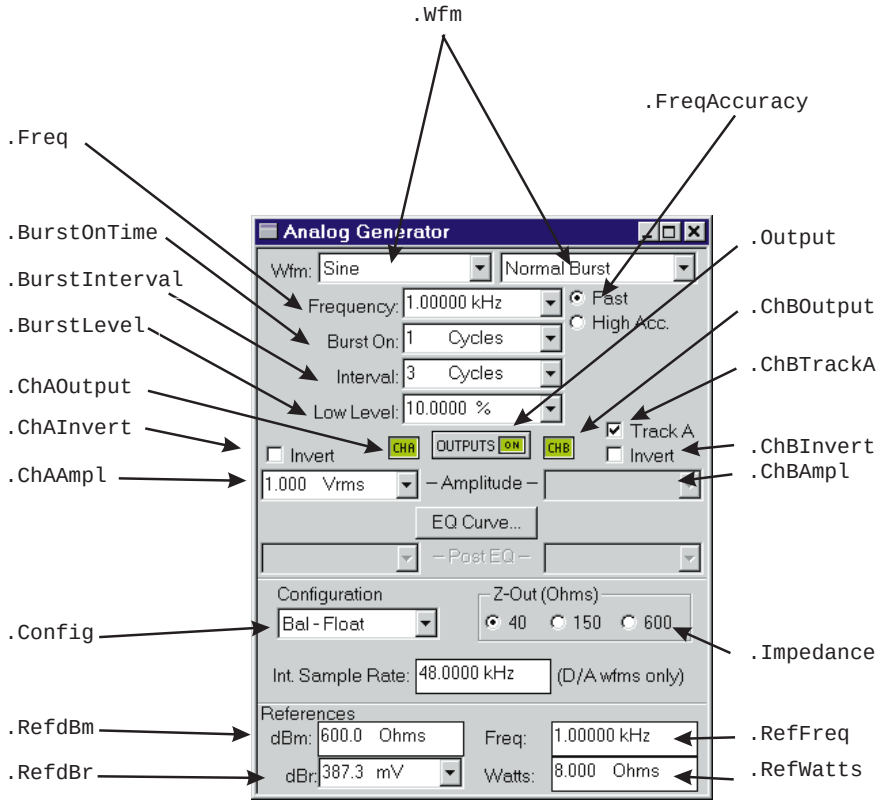
❷ Analog Analyzer

All commands on this page start with the following:
AP.Anlr
Example: AP.Anlr.ChACoupling



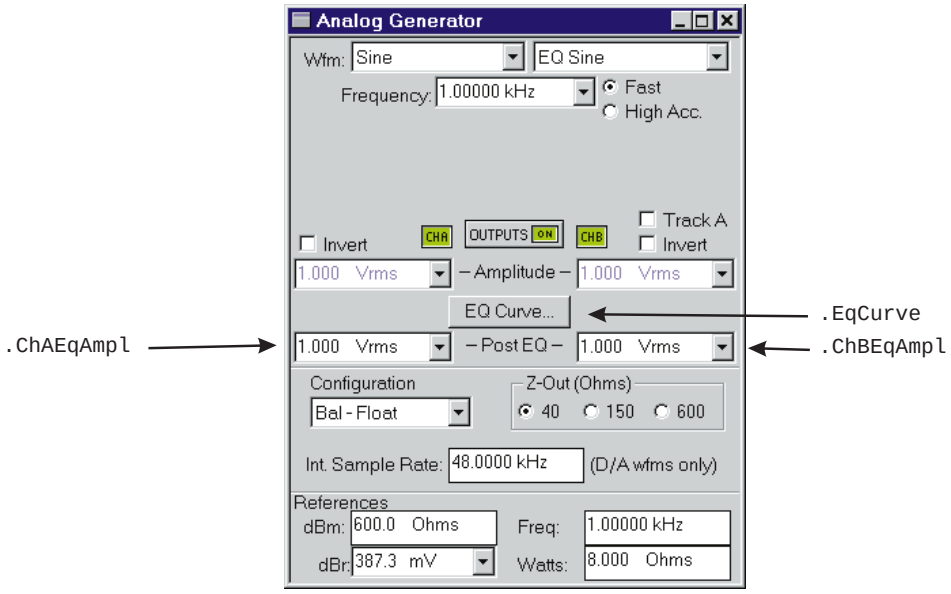
❷ Analog Generator ...

All commands on this page start with the following:
AP.Gen
Example: AP.Gen.Freq



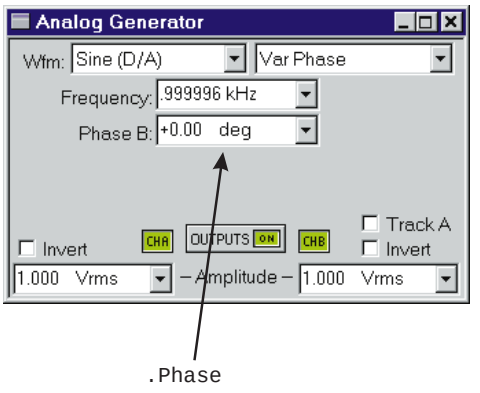
❷ Analog Generator Continued ...

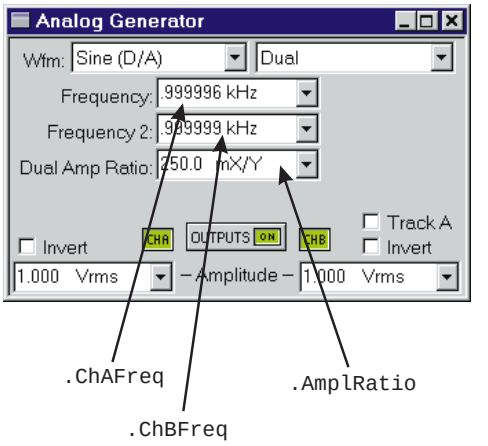
All commands on this page start with the following:
AP.Gen
Example: AP.Gen.IMFreq

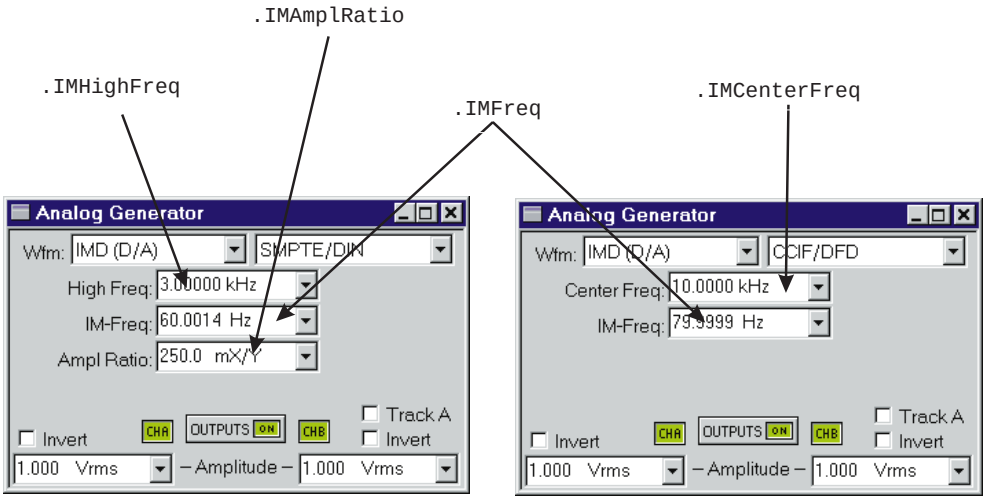


❷ Analog Generator Continued

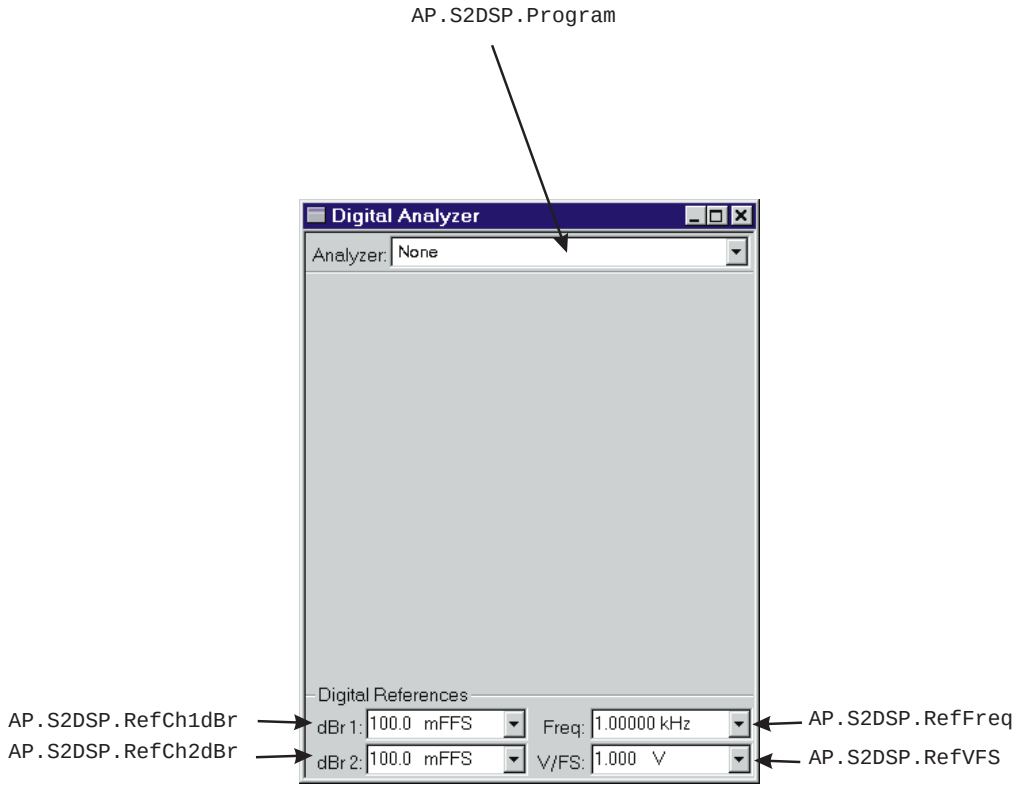
All commands on this page start with the following:
AP.Gen
Example: AP.Gen.Phase





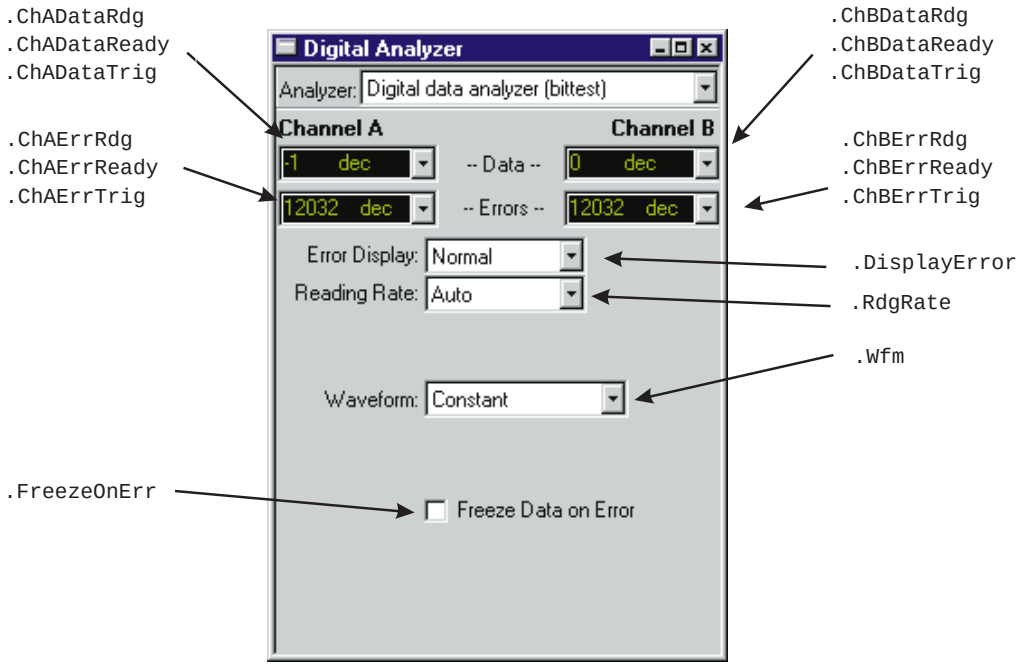


❷ Digital Analyzer Panels



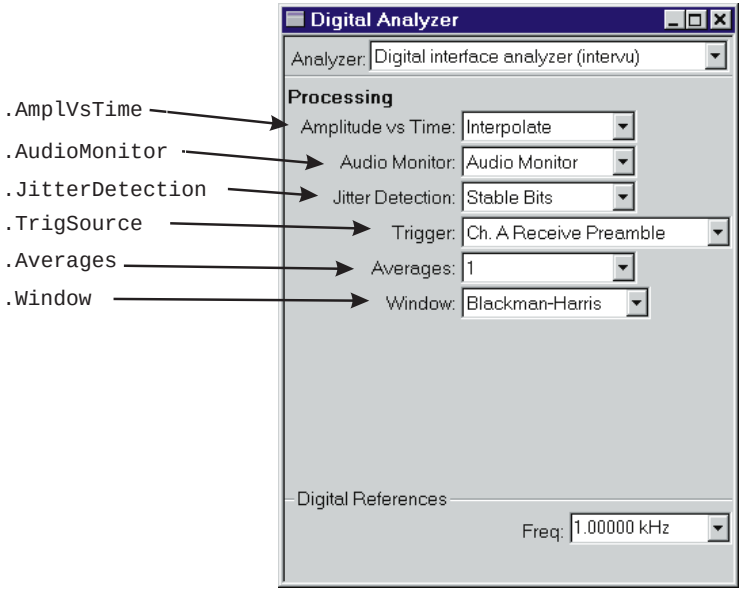
❷ Digital Data Analyzer (BITTEST)

All commands on this page start with the following:
AP.S2DSP.Bittest
Example: AP.S2DSP.Bittest.ChADataRdg



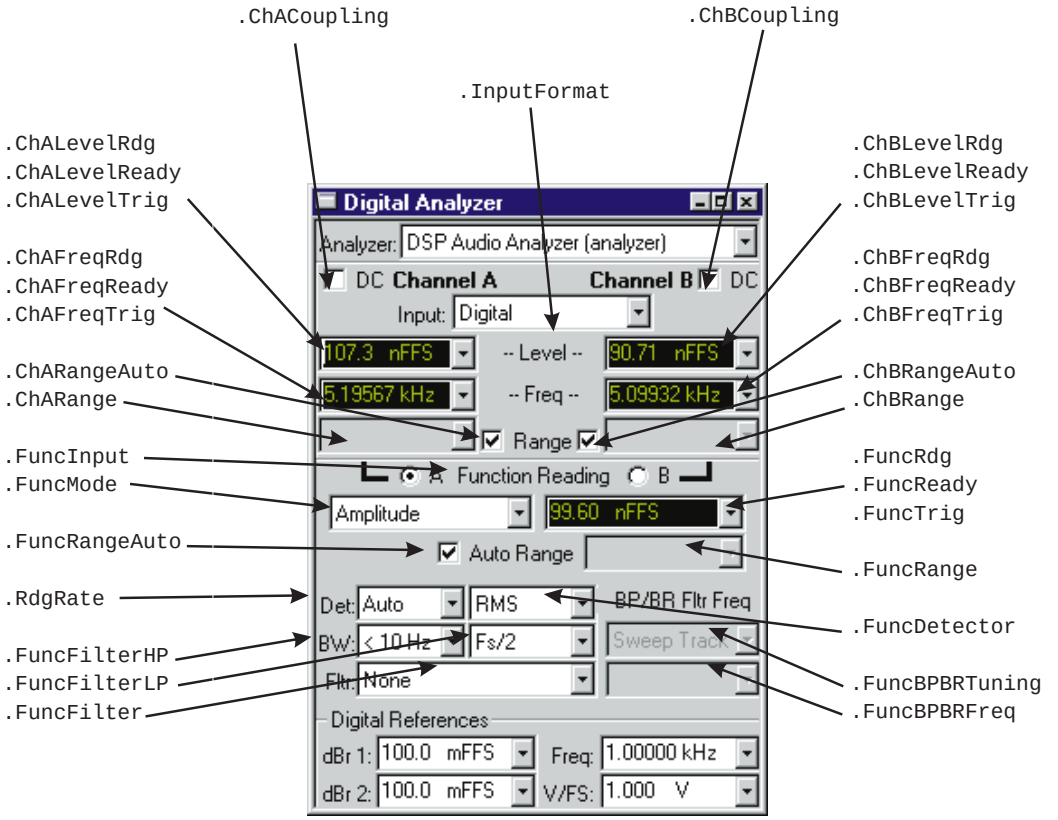
❷ Digital Interface Analyzer (INTERVU)

All commands on this page start with the following:
AP.S2DSP.Intervu
Example: AP.S2DSP.Intervu.AmplVsTime



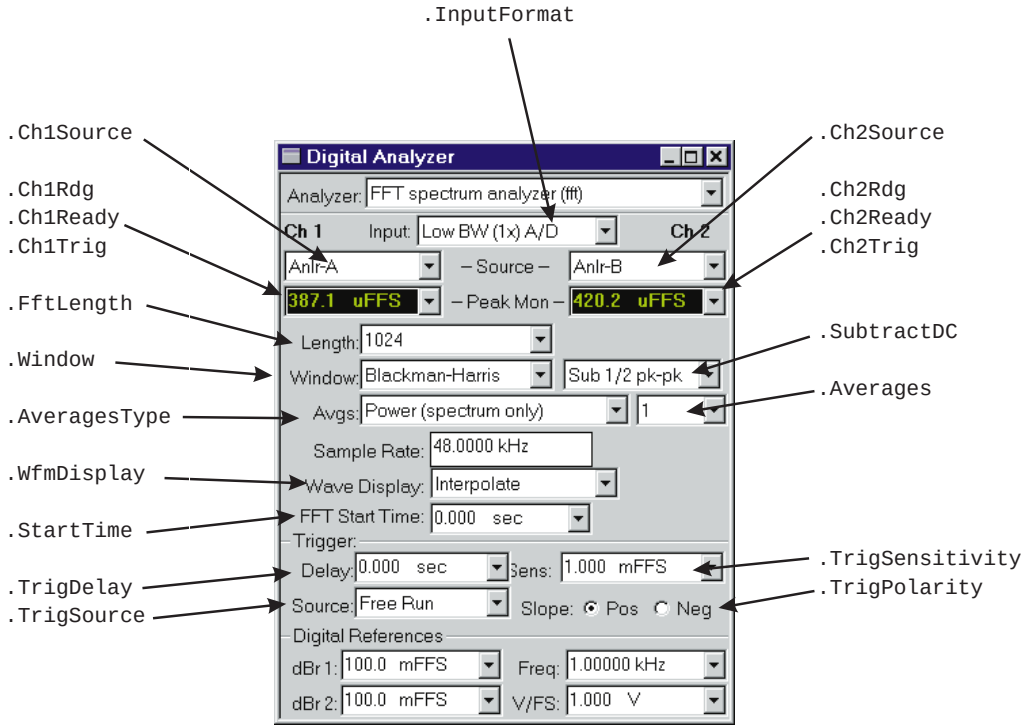
❷ DSP Audio Analyzer (ANALYZER)

All commands on this page start with the following:
AP.S2DSP.Analyzer
Example: AP.S2DSP.Analyzer.ChALevelRdg



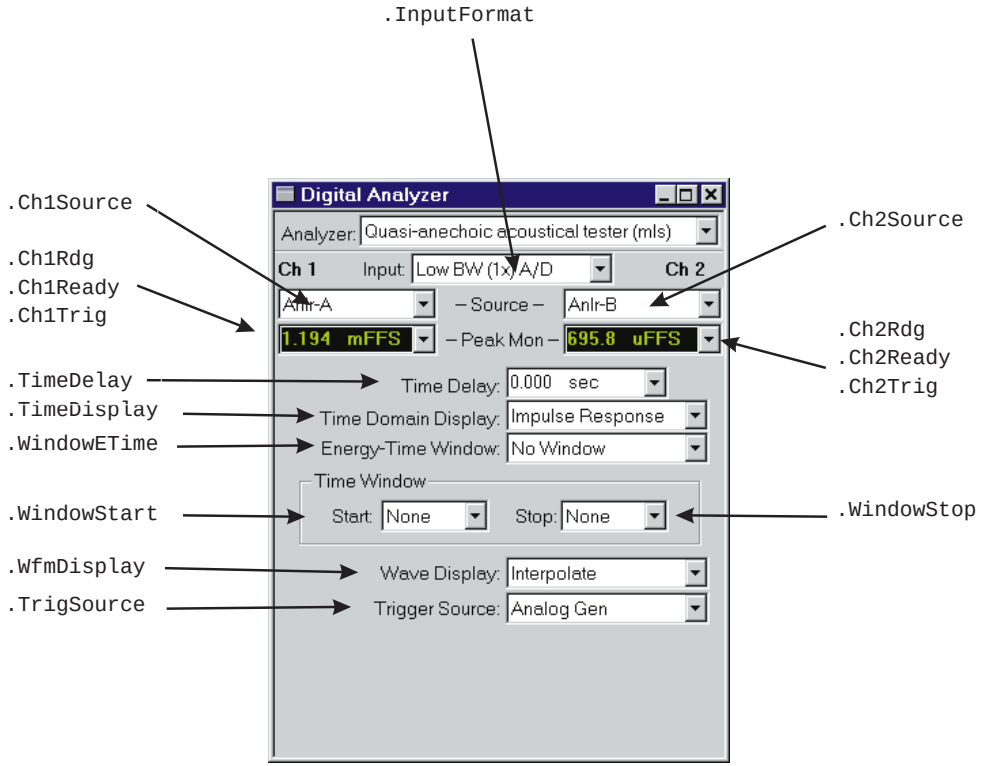
❷ FFT Spectrum Analyzer (FFT)

All commands on this page start with the following:
AP.S2DSP.FFT
Example: AP.S2DSP.FFT.InputFormat



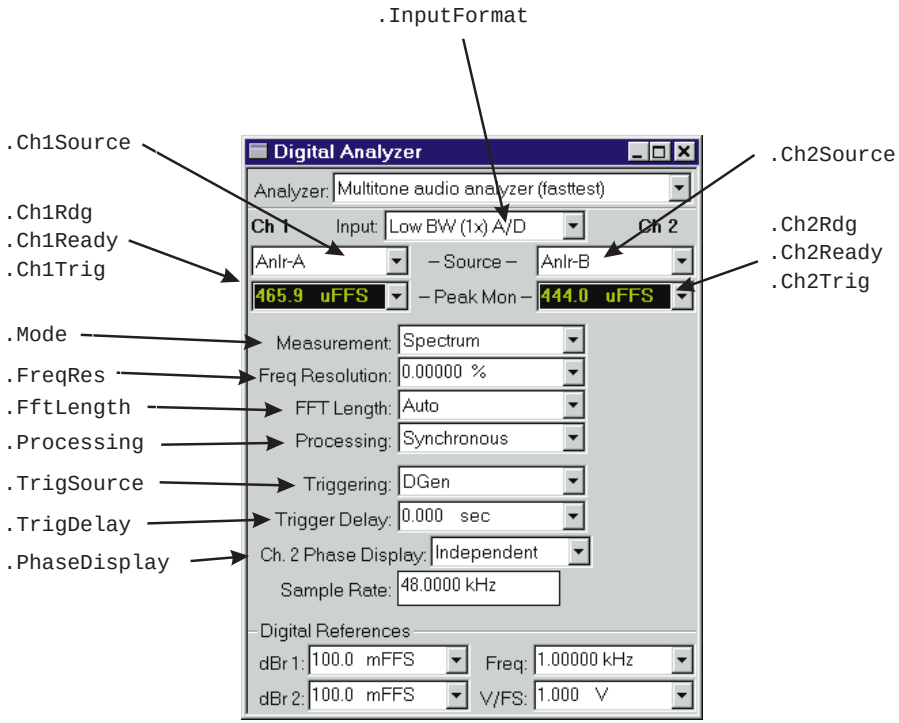
❷ Quasi-Anechoic Acoustical Tester (MLS)

All commands on this page start with the following:
AP.S2DSP.Mls
Example: AP.S1DSP.Mls.InputFormat



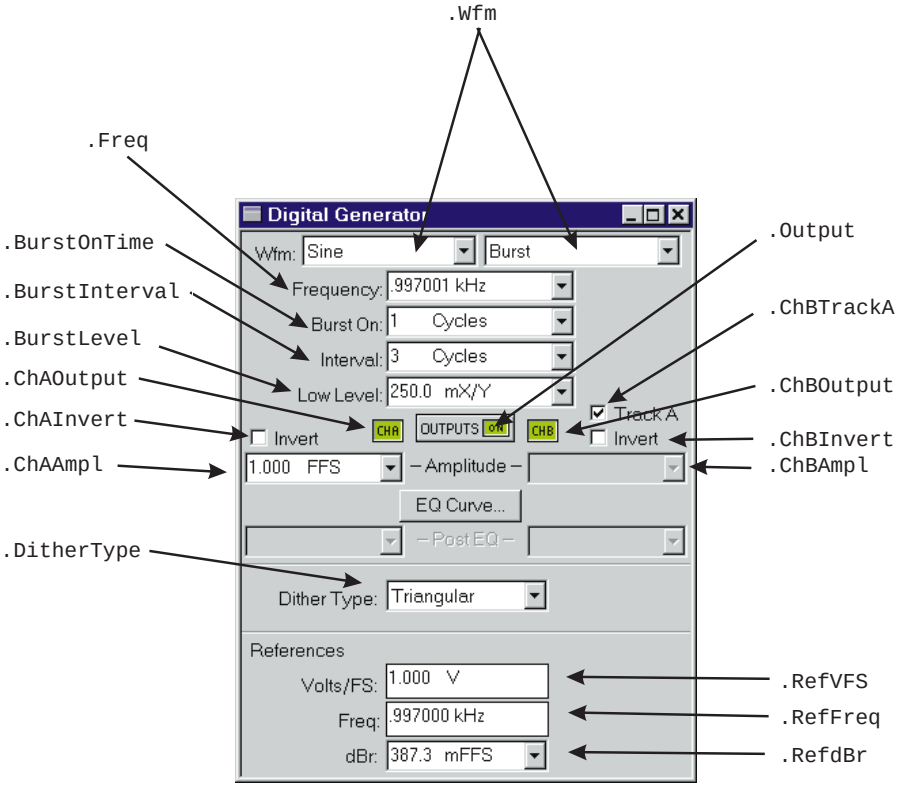
❷ Multitone Audio Analyzer (FASTTEST)

All commands on this page start with the following:
AP.S2DSP.FastTest
Example: AP.S2DSP.FastTest.InputFormat



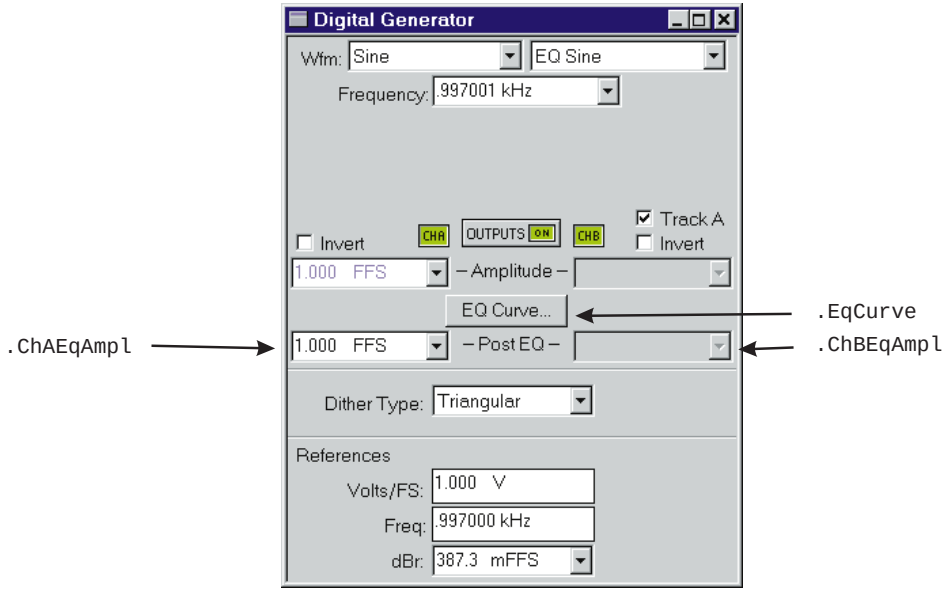
❷ Digital Generator ...

All commands on this page start with the following:
AP.Dgen
Example: AP.Dgen.Freq



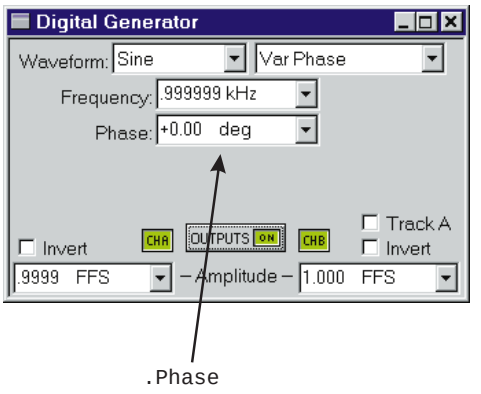
❷ Digital Generator Continued ...

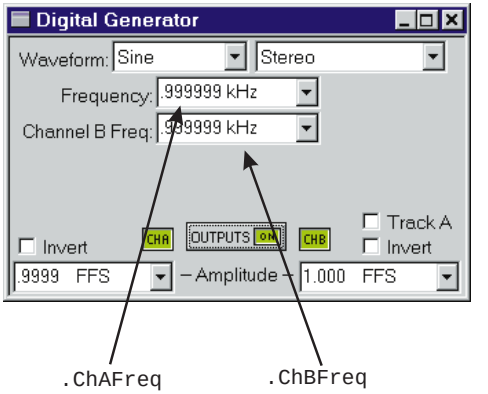
All commands on this page start with the following:
AP.Dgen
Example: AP.Dgen.ChAEqAmpl

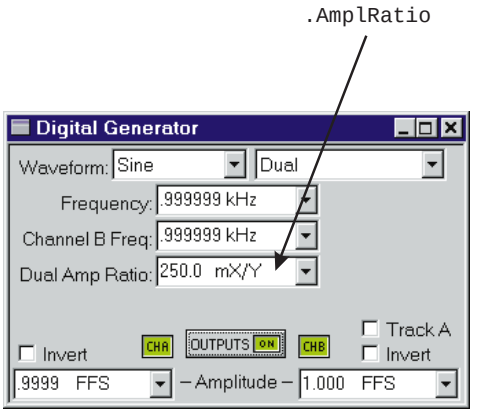


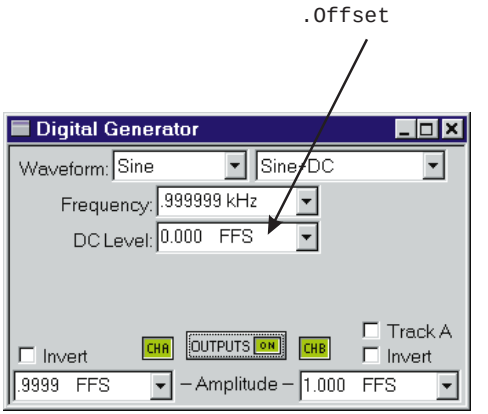
❷ Digital Generator Continued ...

All commands on this page start with the following:
AP.Dgen
Example: AP.Dgen.phase



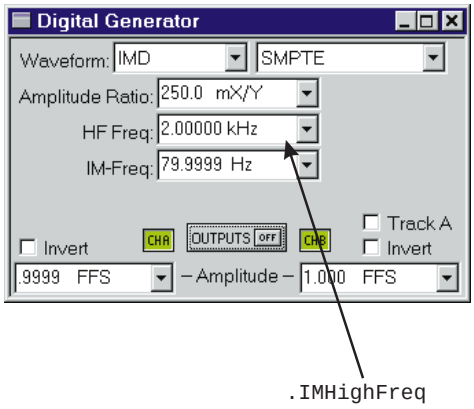


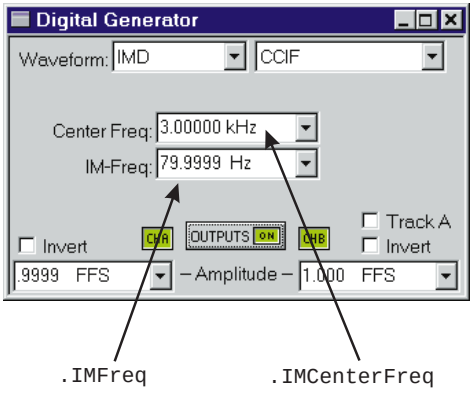


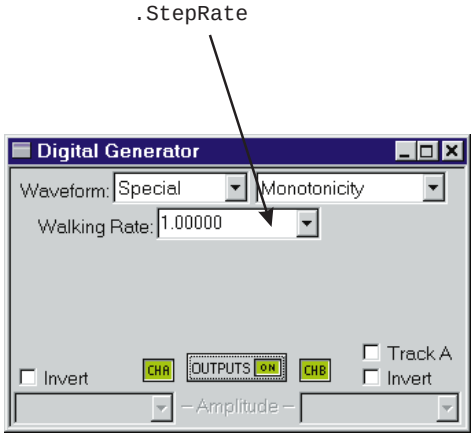


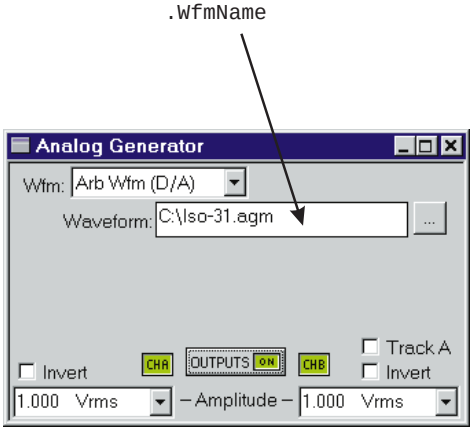
❷ Digital Generator Continued

All commands on this page start with the following:
AP.Dgen
Example: AP.Dgen.IMFreq



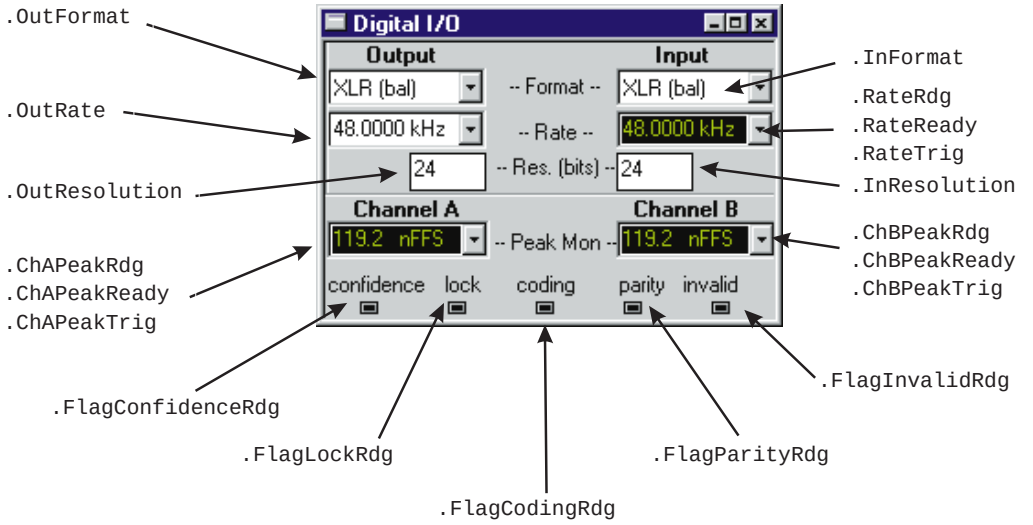






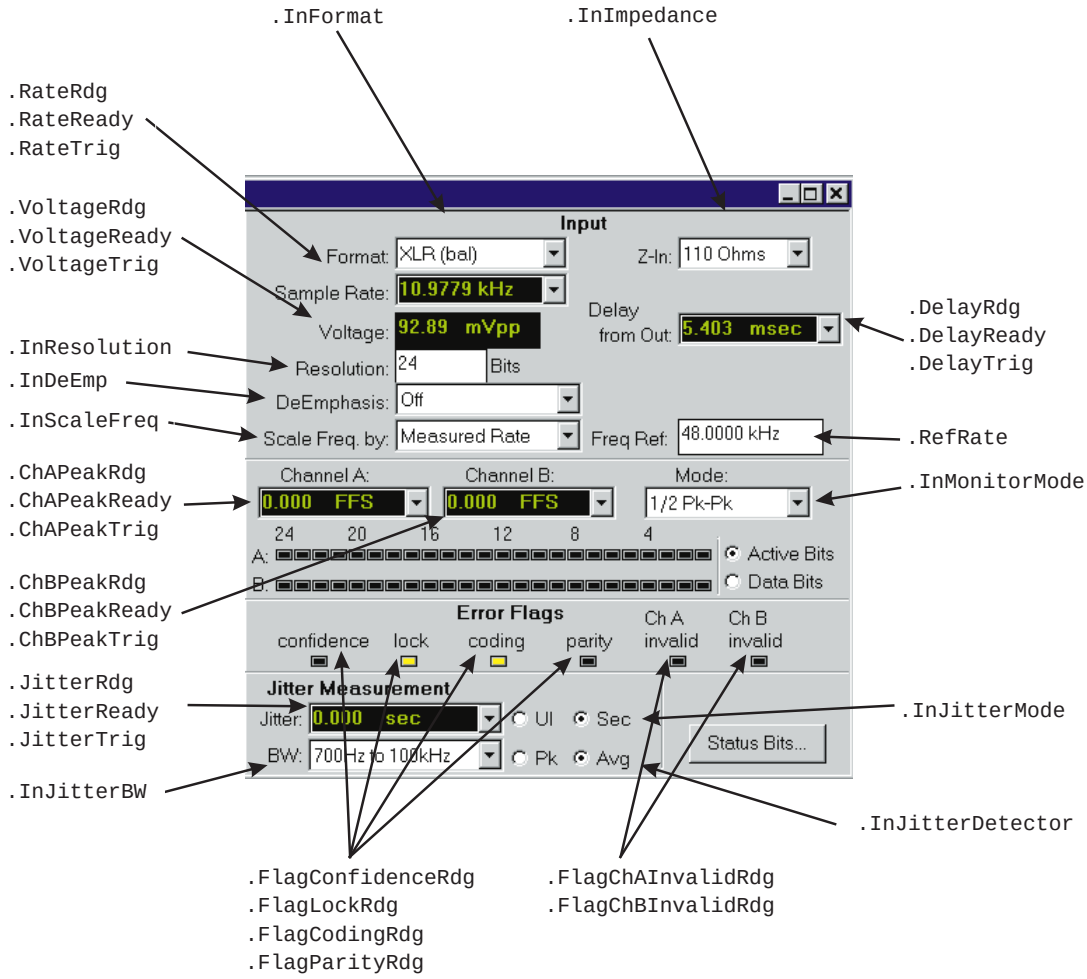
❷ Digital IO Parameters - Input/Output small panel view...

All commands on this page start with the following:
AP.S2Dio
Example: AP.S2Dio.OutFormat



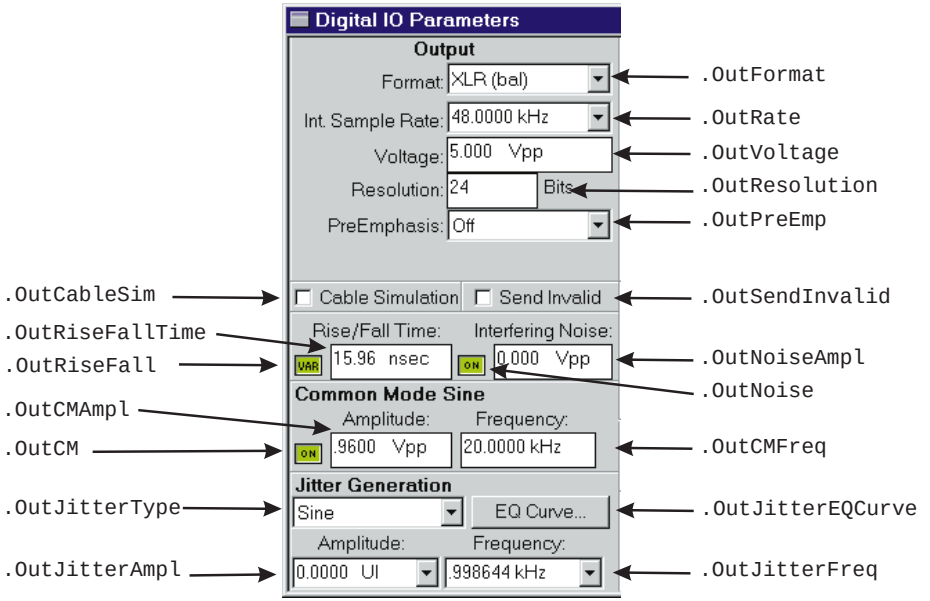
Digital IO Parameters - Input ...

All commands on this page start with the following:
AP.S2Dio
Example: AP.S2Dio.InFormat



❷ Digital IO Parameters - Output Continued

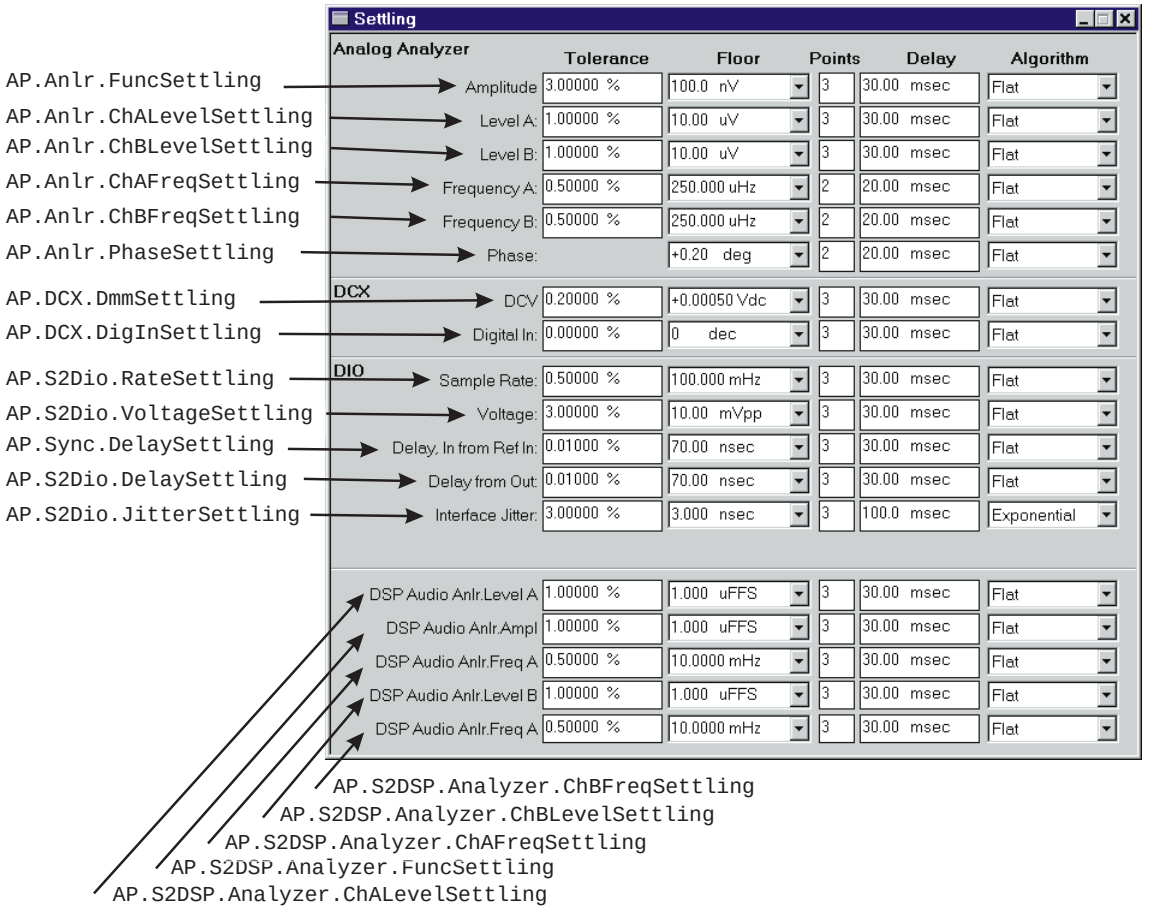
All commands on this page start with the following:
AP.S2Dio
Example: AP.S2Dio.OutFormat



❷ Speaker

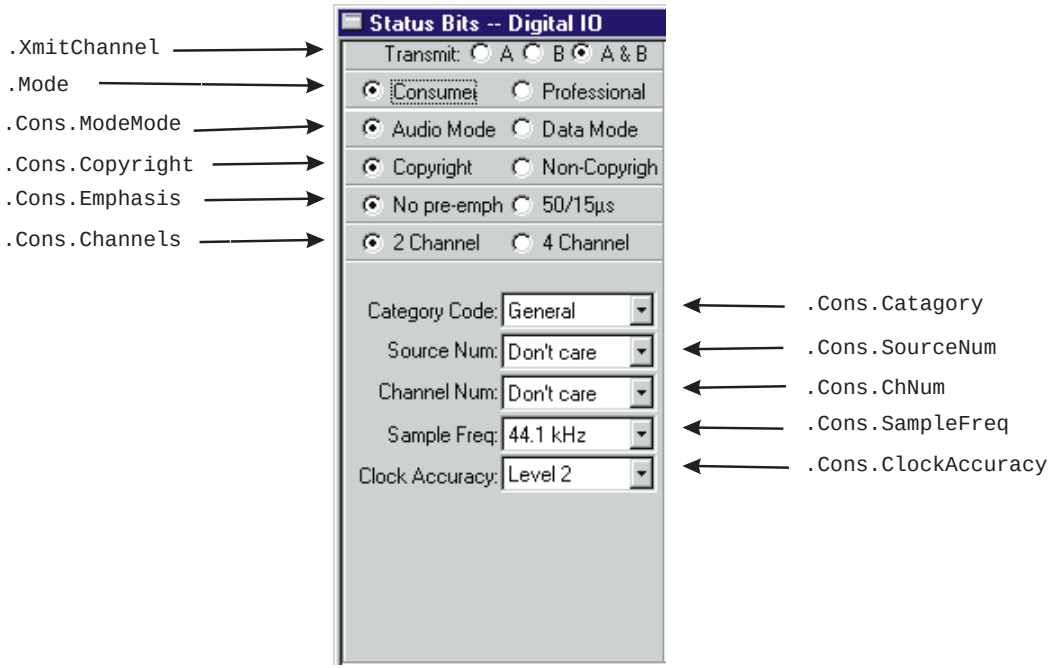
All commands on this page start with the following:
AP.Dgen
Example: AP.Speaker.Source

❷ Settling


Settling
Analog Analyzer
Tolerance
Floor
Points
Delay
Algorithm
AP.Anlr.FuncSettling
Amplitude 3.00000 % 100.0 nV 3 30.00 msec Flat
AP.Anlr.ChALevelSettling
Level A: 1.00000 % 10.00 uV 3 30.00 msec Flat
AP.Anlr.ChBLevelSettling
Level B: 1.00000 % 10.00 uV 3 30.00 msec Flat
AP.Anlr.ChAFreqSettling
Frequency A: 0.50000 % 250.000 uHz 2 20.00 msec Flat
AP.Anlr.ChBFreqSettling
Frequency B: 0.50000 % 250.000 uHz 2 20.00 msec Flat
AP.Anlr.PhaseSettling
Phase: +0.20 deg 2 20.00 msec Flat
DCX
AP.DCX.DmmSettling
DCV 0.20000 % +0.00050 Vdc 3 30.00 msec Flat
AP.DCX.DigInSettling
Digital In: 0.00000 % 0 dec 3 30.00 msec Flat
DIO
AP.S2Dio.RateSettling
Sample Rate: 0.50000 % 100.000 mHz 3 30.00 msec Flat
AP.S2Dio.VoltageSettling
Voltage: 3.00000 % 10.00 mVpp 3 30.00 msec Flat
AP.Sync.DelaySettling
Delay, In from Ref In: 0.01000 % 70.00 nsec 3 30.00 msec Flat
AP.S2Dio.DelaySettling
Delay from Out: 0.01000 % 70.00 nsec 3 30.00 msec Flat
AP.S2Dio.JitterSettling
Interface Jitter: 3.00000 % 3.000 nsec 3 100.0 msec Exponential
DSP Audio Anlr.Level A 1.00000 % 1.000 uFFS 3 30.00 msec Flat
DSP Audio Anlr.Ampl 1.00000 % 1.000 uFFS 3 30.00 msec Flat
DSP Audio Anlr.Freq A 0.50000 % 10.0000 mHz 3 30.00 msec Flat
DSP Audio Anlr.Level B 1.00000 % 1.000 uFFS 3 30.00 msec Flat
DSP Audio Anlr.Freq A 0.50000 % 10.0000 mHz 3 30.00 msec Flat
AP.S2DSP.Analyzer.ChBFreqSettling
AP.S2DSP.Analyzer.ChBLevelSettling
AP.S2DSP.Analyzer.ChAFreqSettling
AP.S2DSP.Analyzer.FuncSettling
AP.S2DSP.Analyzer.ChALevelSettling

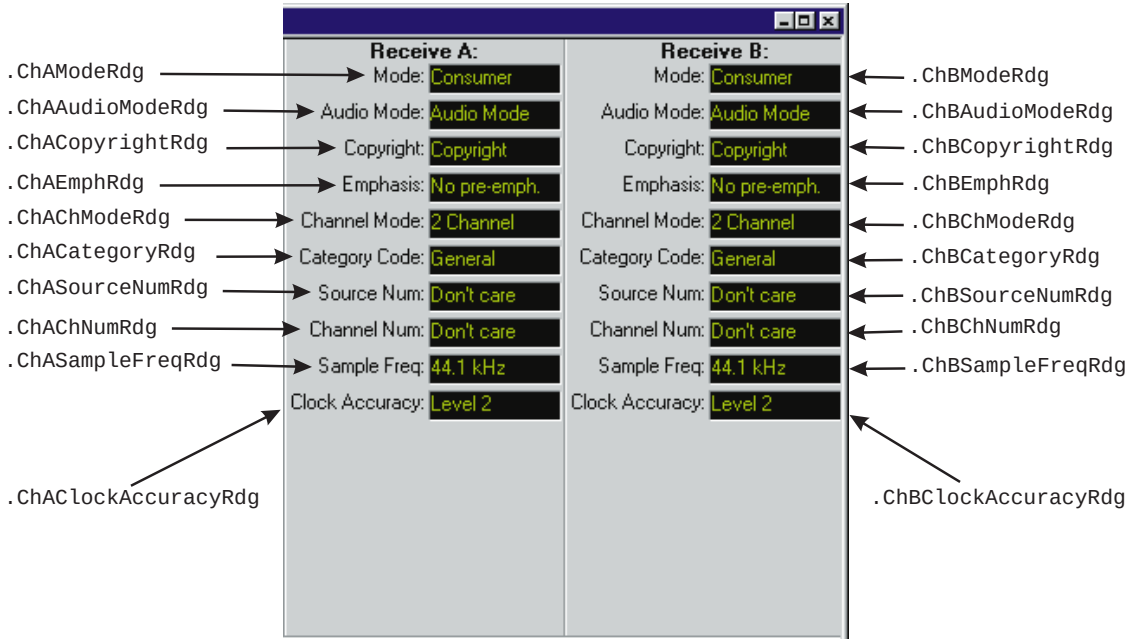
❷ Status Bits — Digital IO - Transmit Consumer

All commands on this page start with the following:
AP.Bits
Example: AP.Bits.Mode



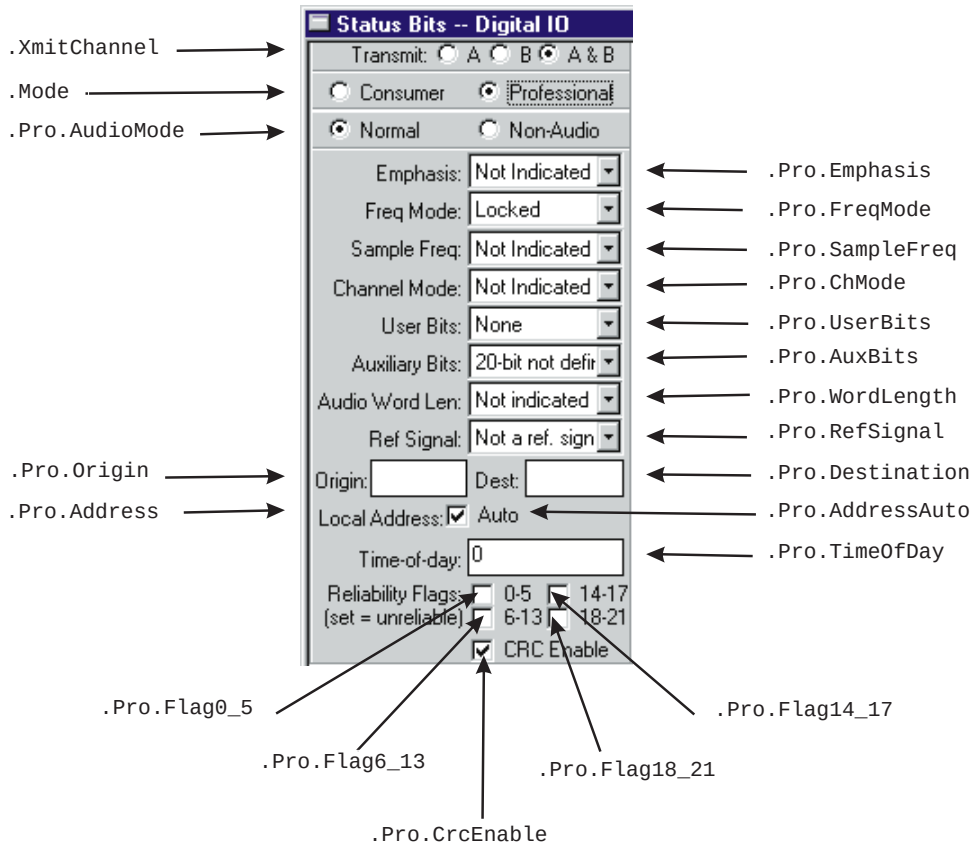
❷ Status Bits — Digital IO - Receive Consumer

All commands on this page start with the following:
AP.Bits
Example: AP.Bits.

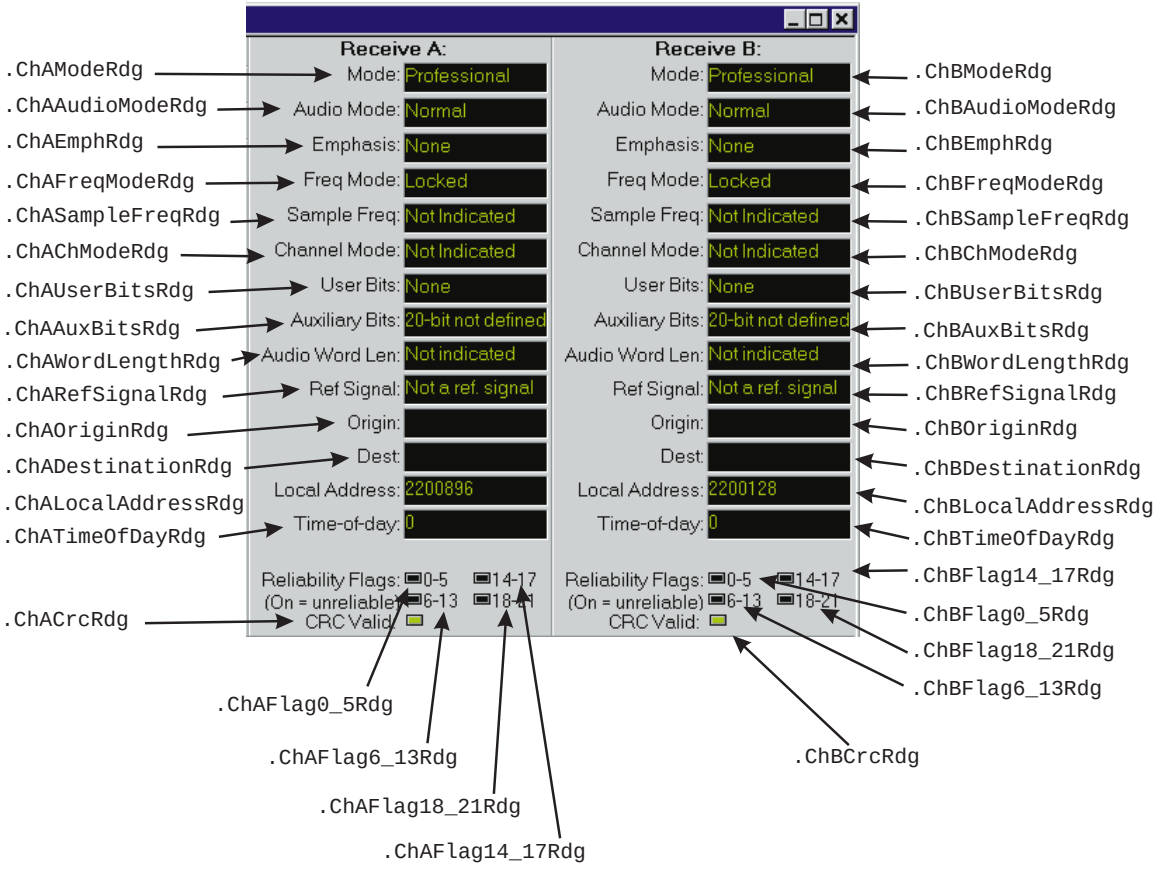


❷ Status Bits — Digital IO - Transmit Professional

All commands on this page start with the following:
AP.Bits
Example: AP.Bits.XmitChannel

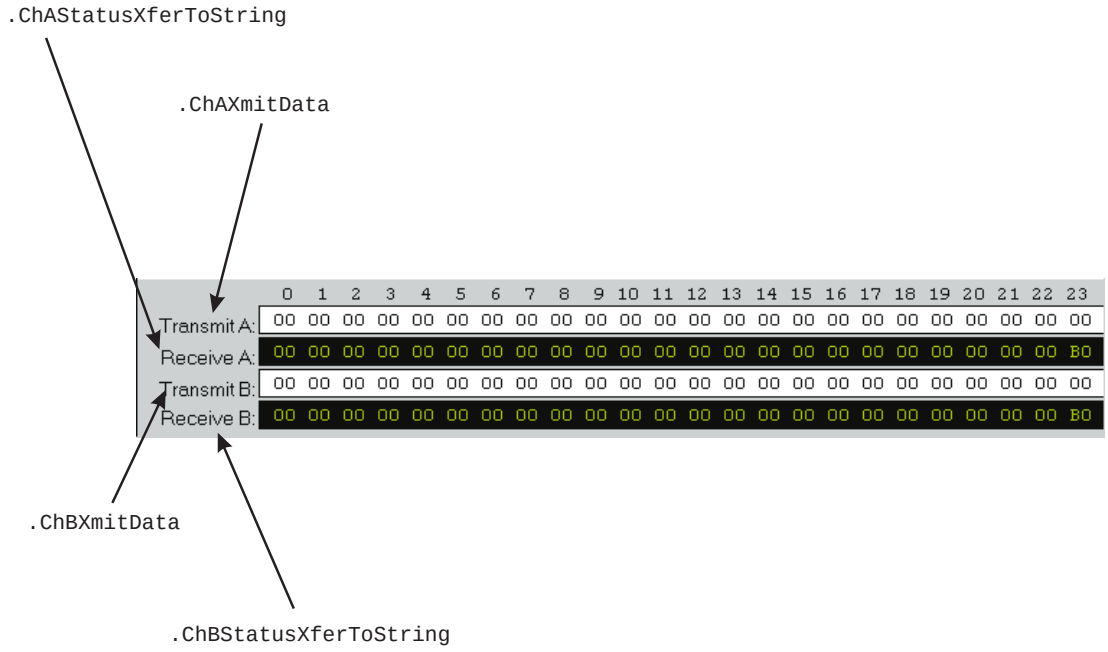


All commands on this page start with the following:
AP.Bits
Example: AP.Bits.



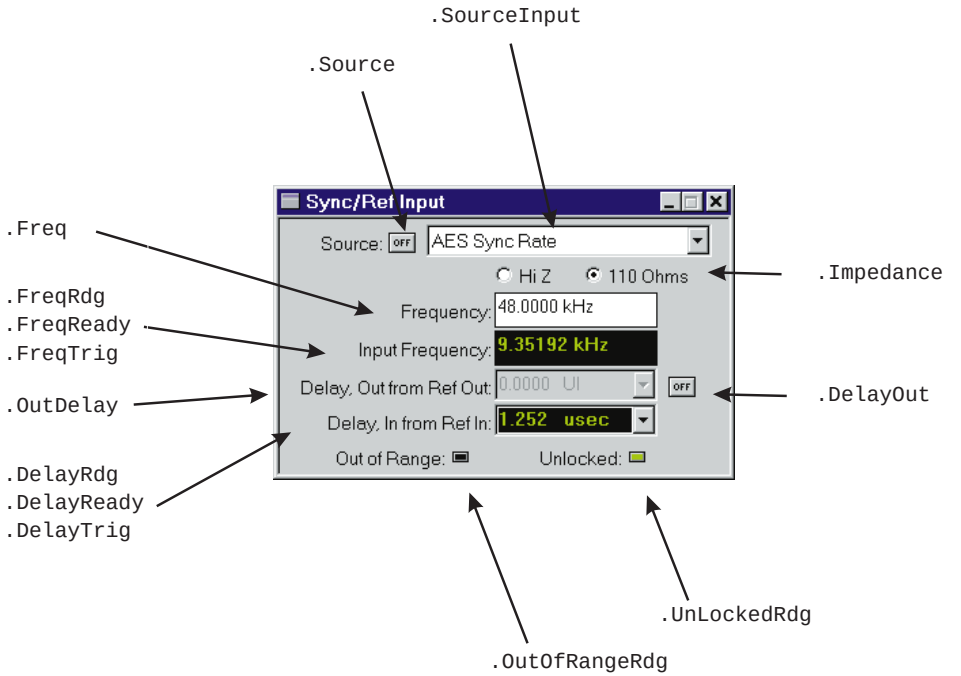
❷ Status Bits — Digital IO - Receive Professional

All commands on this page start with the following:
AP.Bits
Example: AP.Bits.ChAStatusXferToArray



❷ Sync

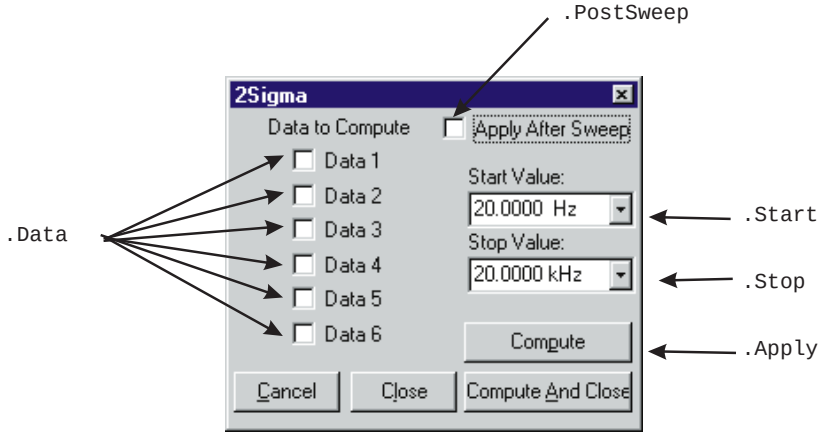
All commands on this page start with the following:
AP.Sync
Example: AP.Sync.Source



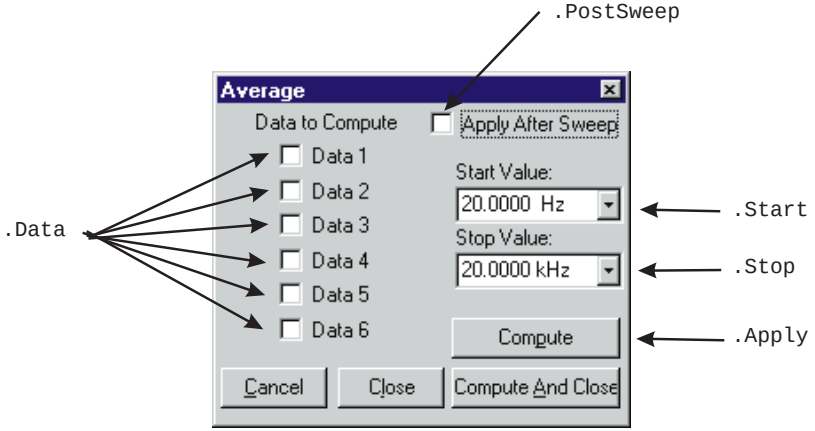
System One and System Two Panels

❶❷ Computes ...

All commands for the top diagram start with the following:
AP.Compute.Sigma
Example: AP.Compute.Sigma.Apply

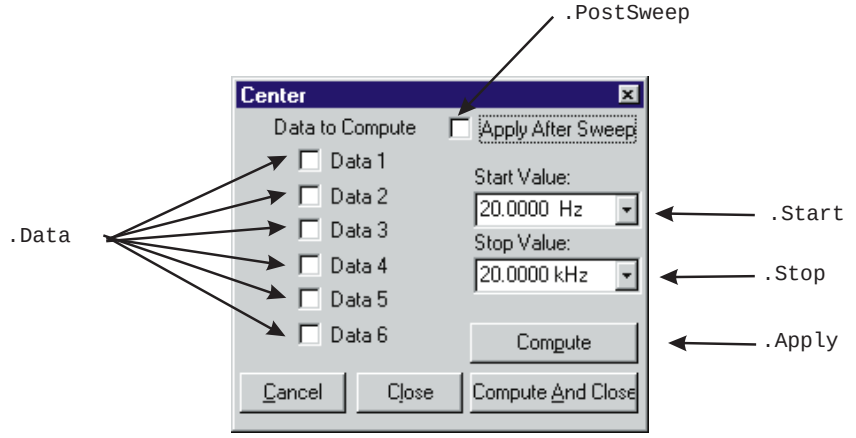


All commands for the top diagram start with the following:
AP.Compute.Avg
Example: AP.Compute.Avg.Apply

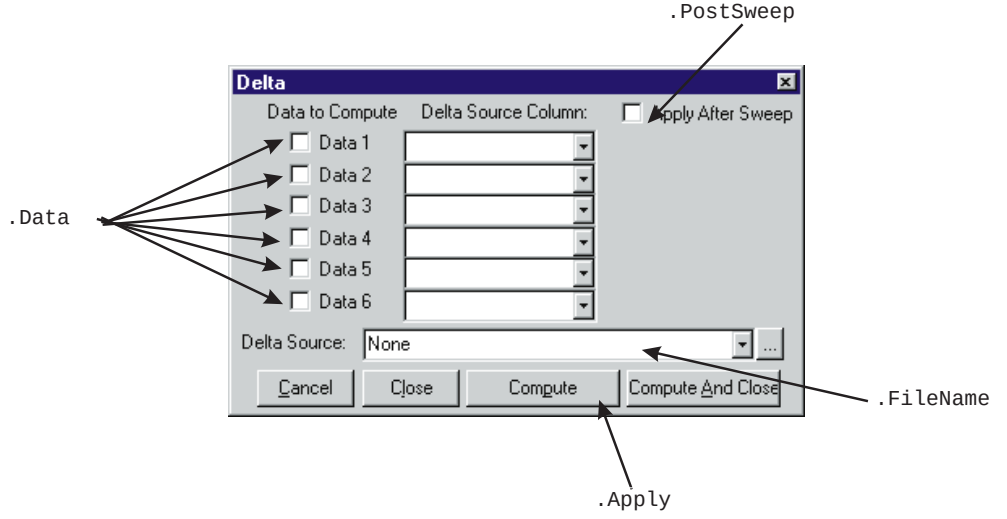


❶❷ Computes Continued ...

All commands for the top diagram start with the following:
AP.Compute.Center
Example: AP.Compute.Center.Apply

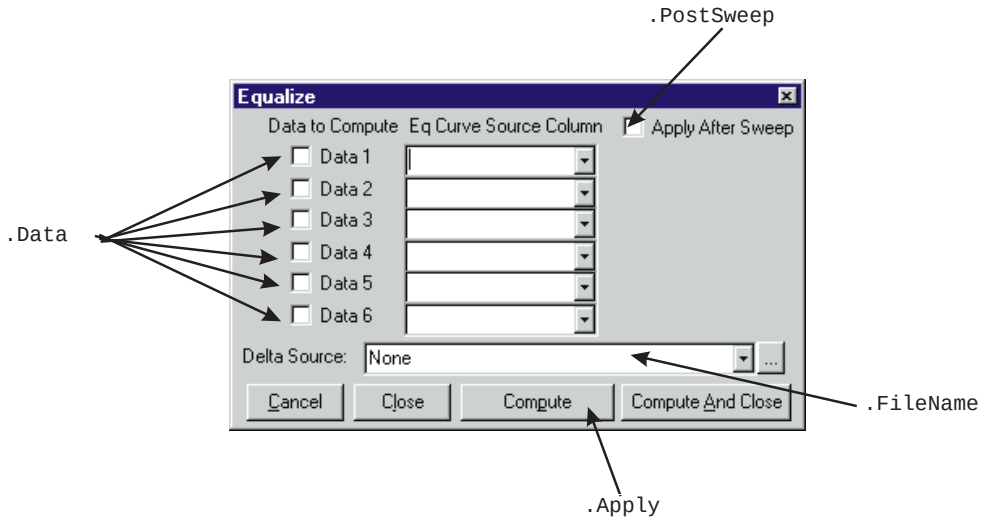


All commands for the top diagram start with the following:
AP.Compute.Delta
Example: AP.Compute.Delta.Apply



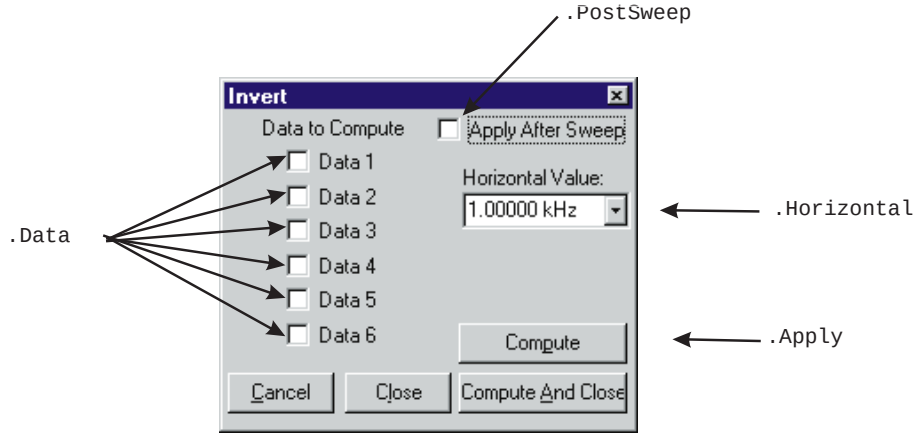
❶❷ Computes Continued ...

All commands for the top diagram start with the following:
AP.Compute.Equalize
Example: AP.Compute.Equalize.Apply

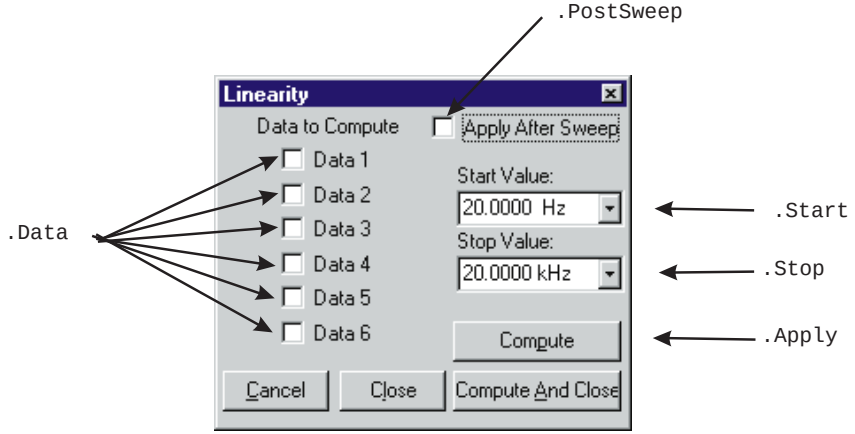


❶❷ Computes Continued ...

All commands for the top diagram start with the following:
AP.Compute.Invert
Example: AP.Compute.Invert.Apply

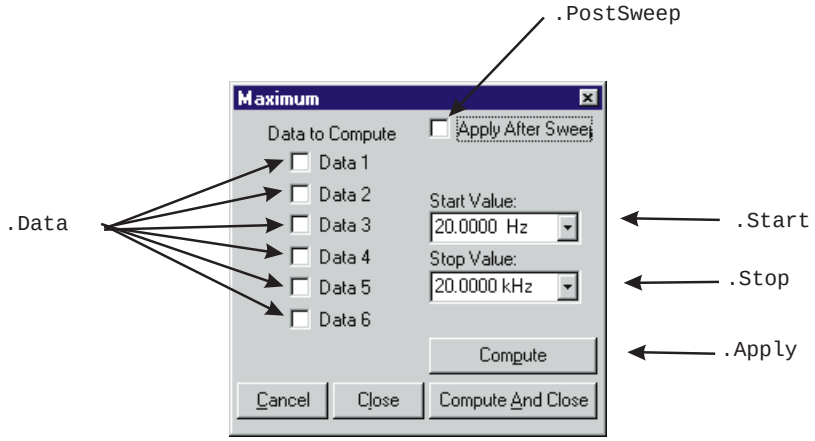


All commands for the top diagram start with the following:
AP.Compute.Linearity
Example: AP.Compute.Linearity.Apply

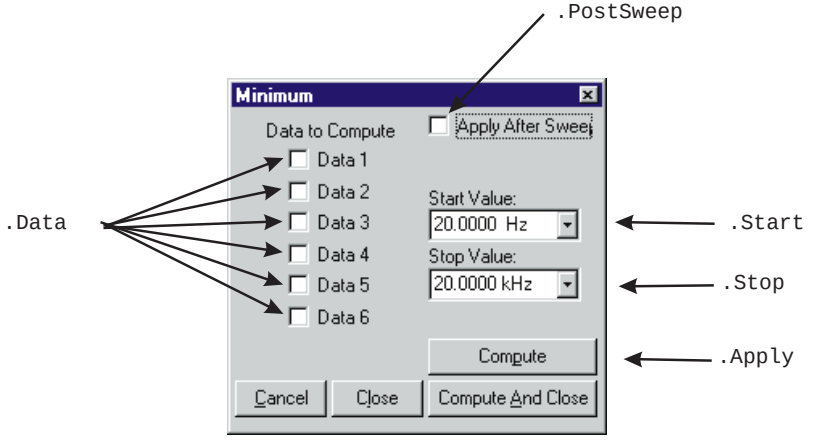


❶❷ Computes Continued ...

All commands for the top diagram start with the following:
AP.Compute.Max
Example: AP.Compute.Max.Apply

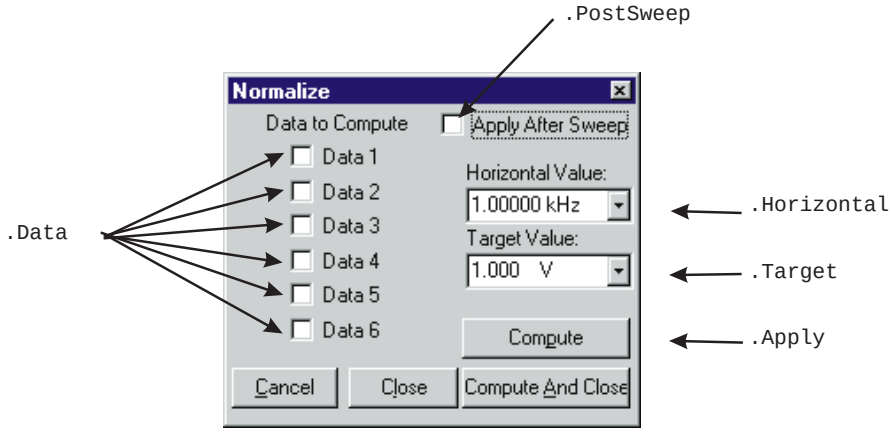


All commands for the top diagram start with the following:
AP.Compute.Min
Example: AP.Compute.Min.Apply

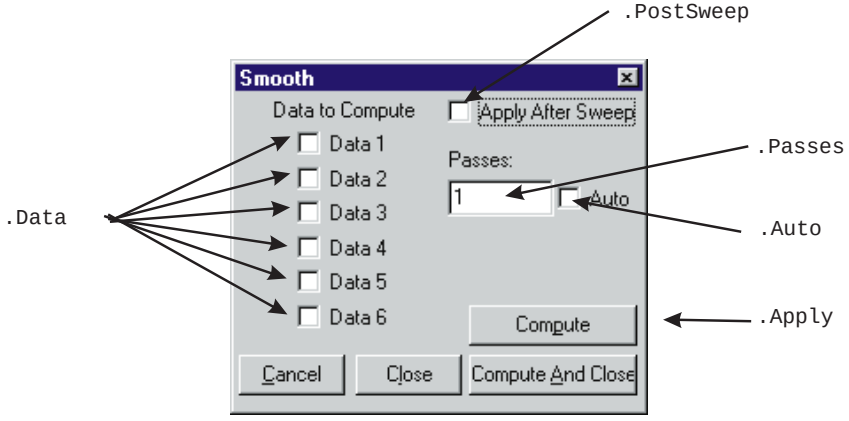


❶❷ Computes Continued ...

All commands for the top diagram start with the following:
AP.Compute.Normalize
Example: AP.Compute.Normalixe.Apply

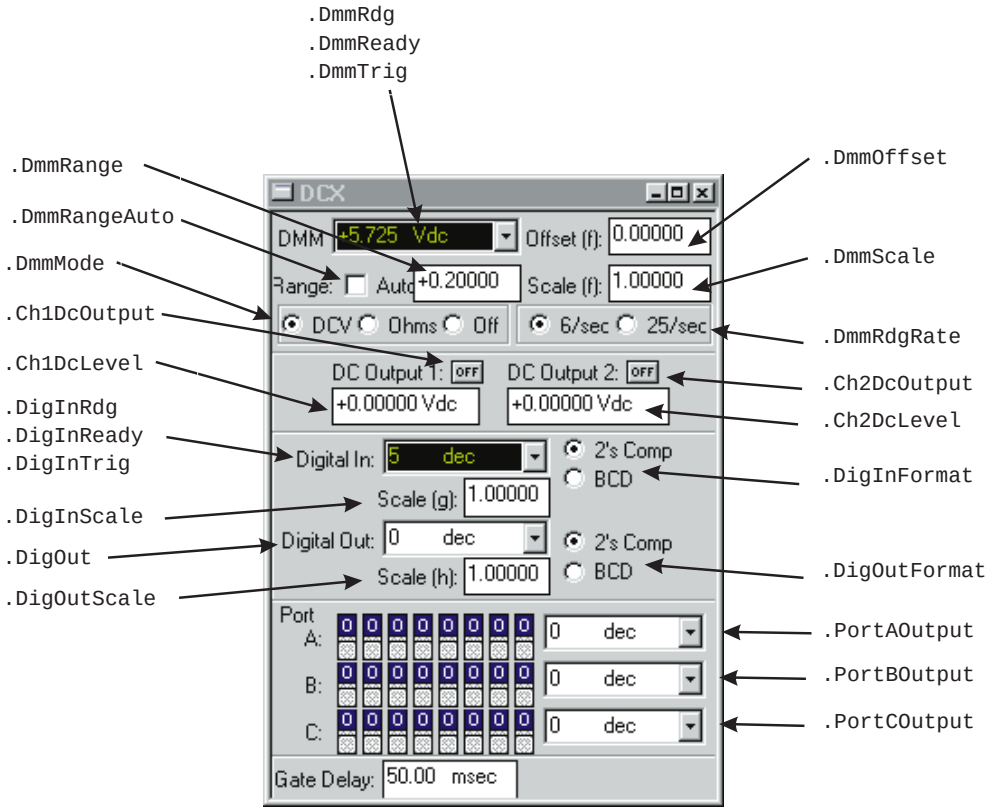


All commands for the top diagram start with the following:
AP.Compute.Smooth
Example: AP.Compute.Smooth.Apply



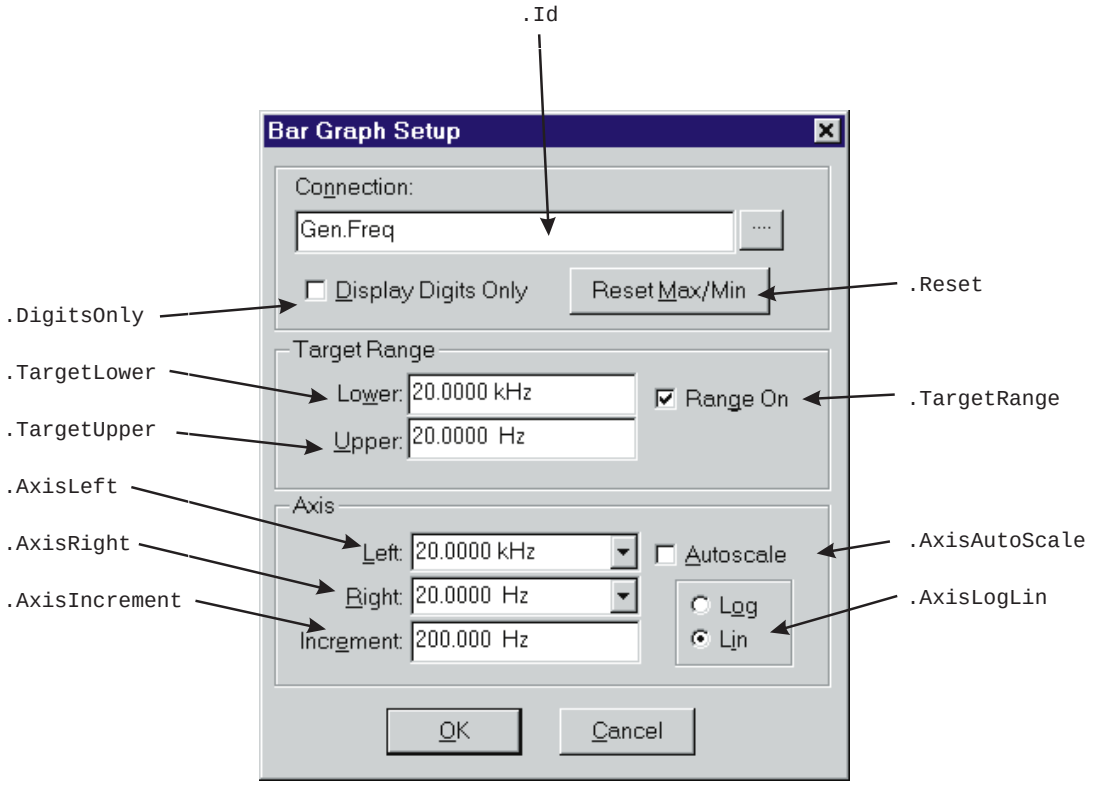
❶❷ DCX-127

All commands on this page start with the following:
AP.DCX
Example: AP.S2DSP.FastTest.InputFormat



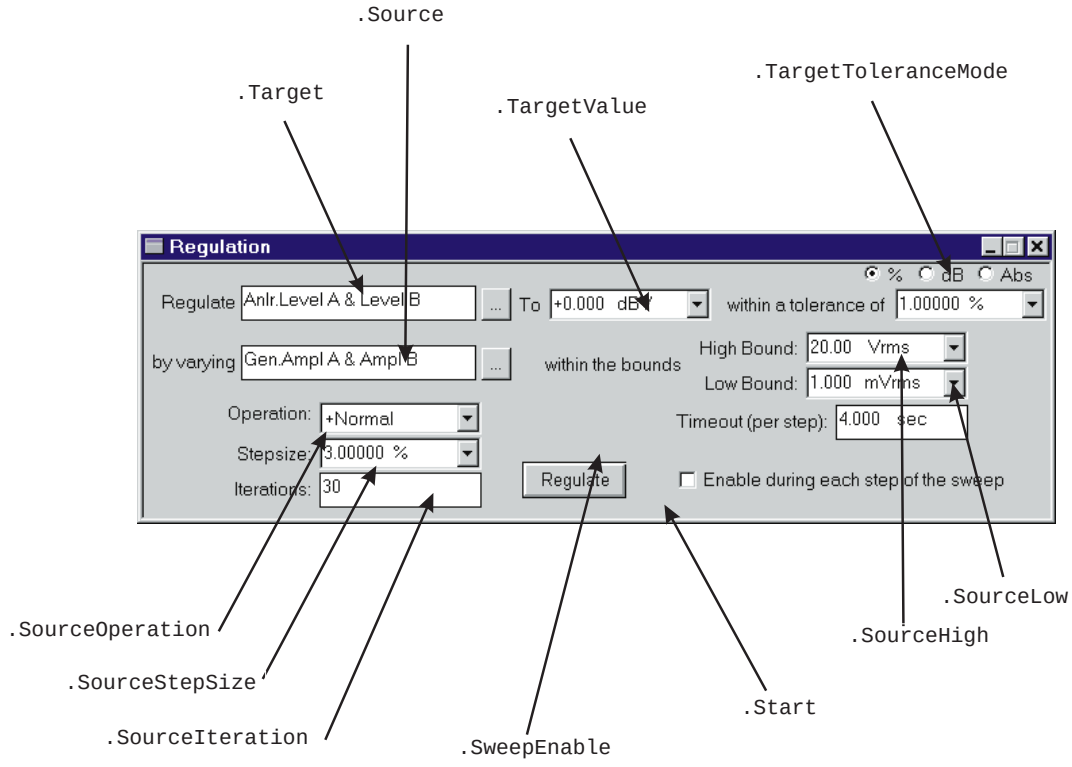
❶❷ Bar Graph

All commands on this page start with the following:
AP.Bar
Example: AP.Bar.FastTest.InputFormat



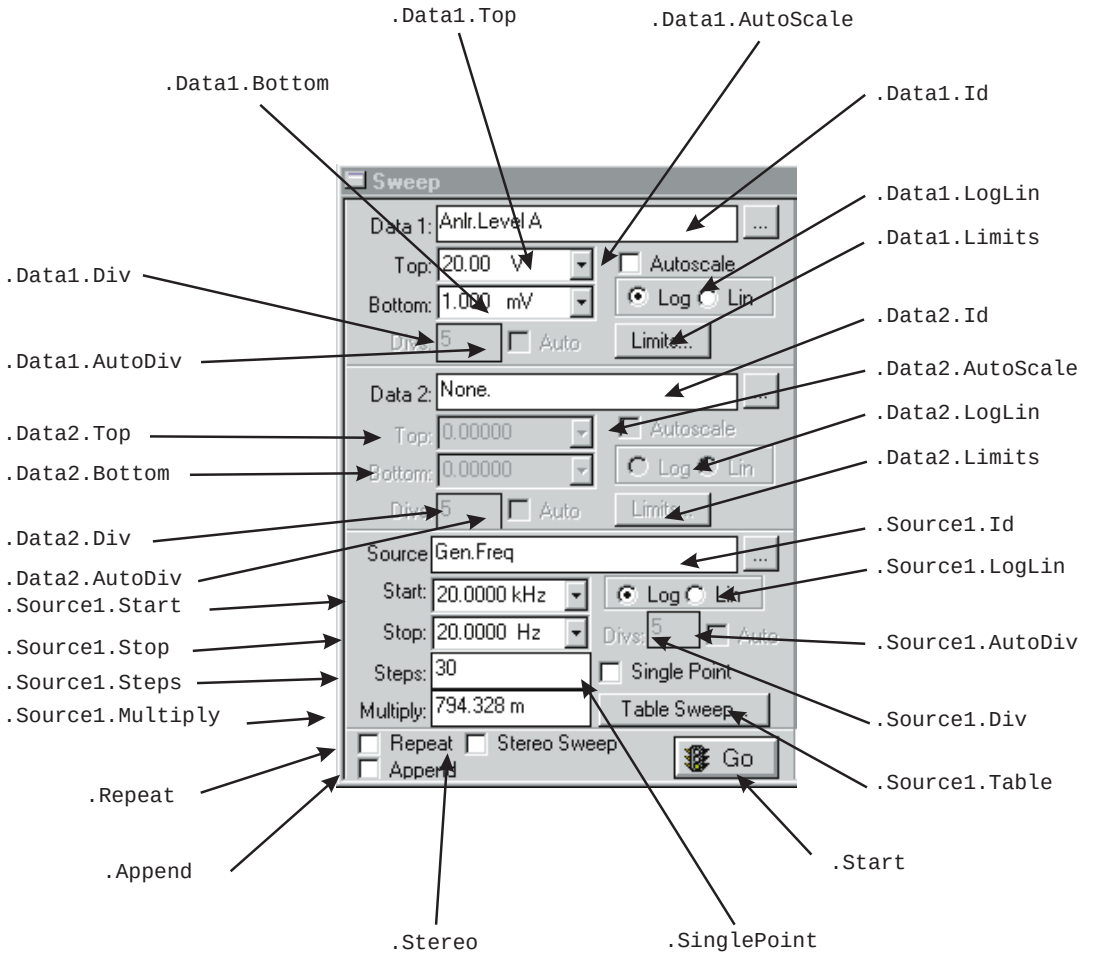
❶❷ Regulation

All commands on this page start with the following:
AP.Reg
Example: AP.Sweep.Data1.Id



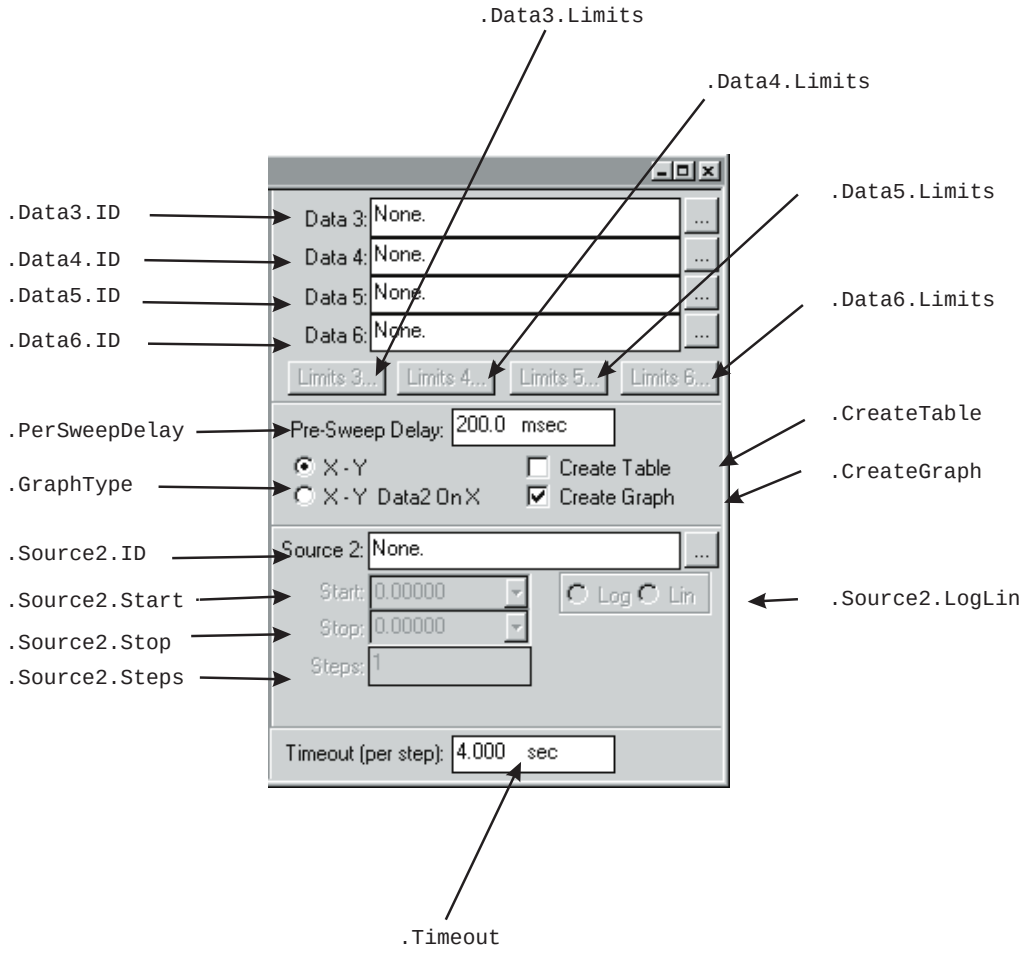
❶❷ Sweep ...

All commands on this page start with the following:
AP.Sweep
Example: AP.Sweep.Data1.Id



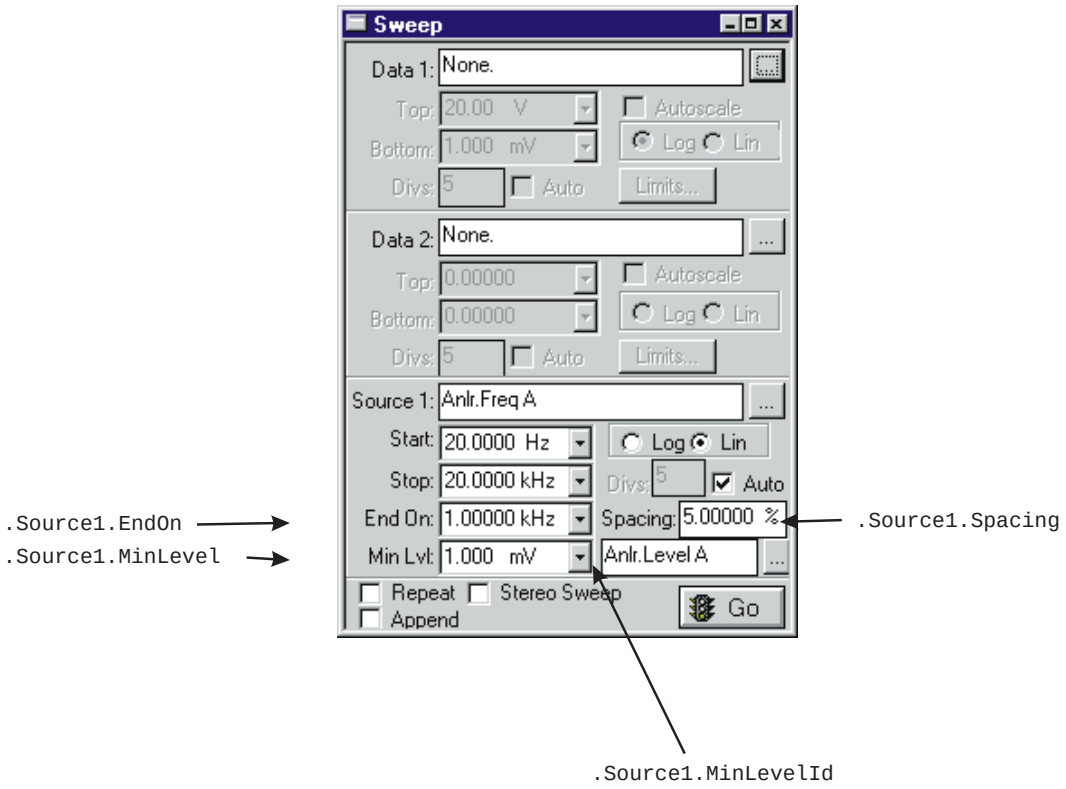
❶❷ Sweep Continued ...

All commands on this page start with the following:
AP.Sweep
Example: AP.Sweep.Data1.Id



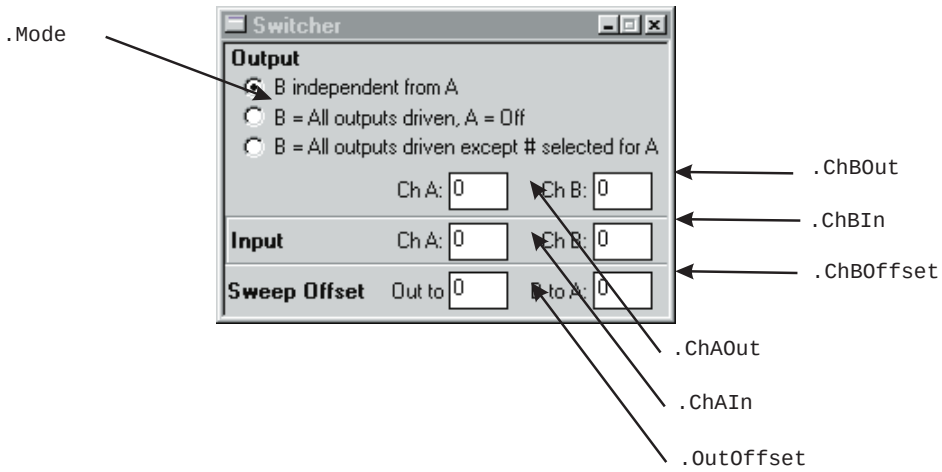
❶❷ Sweep Continued ...

All commands on this page start with the following:
AP.Sweep
Example: AP.Sweep.Data1.Id



❶❷ Switcher

All commands on this page start with the following:
AP.SWR
Example: AP.SWR.Mode



Alphabetical Command Listing by Group

Analyzer

AP.Anlr *	
.ChACoupling	Channel A DC Coupled
.ChAFreqRdg	Channel A Frequency Reading
.ChAFreqReady	Channel A Frequency Ready
.ChAFreqSettling	Channel A Frequency Settling
.ChAFreqTrig	Channel A Frequency Trigger
.ChAImpedance	Channel A Impedance
.ChAInput	Channel A Input
.ChALevelRdg	Channel A Level Reading
.ChALevelReady	Channel A Level Ready
.ChALevelSettling	Channel A Level Settling
.ChALevelTrig	Channel A Level Trigger
.ChARange	Channel A Input Range
.ChARangeAuto	Channel A Range Auto
.ChBCoupling	Channel B DC Coupled
.ChBFreqRdg	Channel B Frequency Reading
.ChBFreqReady	Channel B Frequency Ready
.ChBFreqSettling	Channel B Frequency Settling
.ChBFreqTrig	Channel B Frequency Trigger
.ChBImpedance	Channel B Impedance
.ChBInput	Channel B Input
.ChBLevelRdg	Channel B Level Reading
.ChBLevelReady	Channel B Level Ready
.ChBLevelSettling	Channel B Level Settling
.ChBLevelTrig	Channel B Level Trigger
.ChBRange	Channel B Input Range
.ChBRangeAuto	Channel B Range Auto
.FreqRdg	Frequency Reading
.FreqReady	Frequency Ready
.FreqSettling	Frequency Settling
.FreqTrig	Frequency Trigger
.FuncBPBRFreq	Function Meter Band Pass Band Reject Filter Frequency
.FuncBPBRTuning	Function Meter Band Pass

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

	Band Reject Frequency Tuning
.FuncDetector	Function Meter Detector Type
.FuncFilter	Function Meter Filter
.FuncFilterHP	Function Meter Band Width Limiting HiPass
.FuncFilterLP	Function Meter Band Width Limiting LowPass
.FuncInput	Function Meter Input Channel
.FuncMode	Function Meter Mode
.FuncRange	Function Meter Range
.FuncRangeAuto	Function Meter Range Auto
.FuncRdg	Function Meter Reading
.FuncReady	Function Meter Ready
.FuncSettling	Function Meter Settling
.FuncTrig	Function Meter Trigger
.LevelRdg	Level Reading
.LevelReady	Level Ready
.LevelSettling	Level Settling
.LevelTrig	Level Trigger
.PhaseMode	Phase Mode
.PhaseRdg	Phase Reading
.PhaseReady	Phase Ready
.PhaseSettling	Phase Settling
.PhaseTrig	Phase Trigger
.RdgRate	Analyzer Meters Reading Rate
.RefChAdBr	Channel A dBr Reference
.RefChBdBr	Channel B dBr Reference
.RefdBm	dBm Reference
.RefdBr	Channel A&B dBr Reference
.RefdBrAuto	Auto Set dBr Reference
.RefFreq	Frequency Reference
.RefFreqAuto	Auto Set Frequency Reference
.RefWatts	Watts Reference
.WFDetector	Wow & Flutter Detector Type
.WFFilter	Wow & Flutter Filter

Application

AP.Application *

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.AppDir	Return APWIN Application Directory(OLE)
.CopyPanelToClipboard	Copy Panel that has focus To Clipboard
.DisplayDataOnTestOpen	Display Data On Test Open
.Input	Input from Computer I/O Address
.MacroDir	Return current Macro directory(OLE)
.Name	Returns the APWIN Application Name
.NewData	New Data
.NewMacro	New Macro (OLE)
.NewTest	New Test
.Output	Output to Computer I/O Address
.Page	Select Page
.PanelClose	Panel Close
.PanelOpen	Panel Numer/Show/Size/Position
.Quit	Quit (OLE)
.Restore	Restore Hardware
.SuppressErrorMessages	Global error prompt handling
.SysType	Get System Type
.TestDir	Return current test directory
.TestName	Return current test name
.Visible	Show/Hide APWIN Application
.VisibleAll	View All Panels/Graphs/Data Editor/Bars
.VisibleBarGraphs	View All Bar Graphs
.VisibleDataEditor	View Data Editor
.VisibleGraph	View Graph
.VisibleMacroEditor	View Macro Editor
.VisiblePanels	View All Panels
.WorkingDir	Set Working Directory(OLE)

Bar Graph

AP.BarGraph *

.AxisAutoScale	Axis Autoscale

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.AxisIncrement	Axis Increment
.AxisLeft	Axis Left Value
.AxisLogLin	Axis Log/Lin
.AxisRight	Axis Right Value
.DigitsOnly	Display Digits Only ON/OFF
.Id	Return/Set Connection Id
.Max	Max Reading
.Min	Min Reading
.New	Display New BarGraph
.Reset	Reset Min / Max on one Bar Graph
.TargetLower	Target Lower Value
.TargetRange	Target Range ON/OFF
.TargetUpper	Target Upper Value

Status Bits

AP.Bits *

.ChAAudioModeRdg	Channel A Mode Reading
.ChAAuxBitsRdg	Channel A Auxiliary Bits Reading
.ChACategoryRdg	Channel A Category Reading
.ChAChModeRdg	Channel A Channel Mode Reading
.ChAChNumRdg	Channel A Channel Number Reading
.ChAClockAccuracyRdg	Channel A Clock Accuracy Reading
.ChACopyrightRdg	Channel A Copyright Reading
.ChACrcRdg	Channel A CRC Valid Reading
.ChADestinationRdg	Channel A Destination Reading
.ChAEmphRdg	Channel A Emph Reading
.ChAFlag0_5Rdg	Channel A Reliability Flag 0-5
.ChAFlag6_13Rdg	Channel A Reliability Flag 6-13
.ChAFlag14_17Rdg	Channel A Reliability Flag 14-17
.ChAFlag18_21Rdg	Channel A Reliability Flag 18-21
.ChAFreqModeRdg	Channel A Freq Mode Reading
.ChALocalAddressRdg	Channel A Local Address Reading
.ChAModeRdg	Channel A Mode Reading
.ChAOriginRdg	Channel A Origin Data Reading
.ChARefSignalRdg	Channel A Ref Signal Reading
.ChASampleFreqRdg	Channel A Sample Frequency Reading
.ChASourceNumRdg	Channel A Source Number Reading
.ChAStatusXferToArray	Channel A Status Bits to String

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.ChAStatusXferToString	Channel A Status Bits to String
.ChATimeOfDayRdg	Channel A Time Of Day Reading
.ChAUserBitsRdg	Channel A User Bits Reading
.ChAWordLengthRdg	Channel A Word Length Reading
.ChAXmitData	Transmit Channel A Status
.ChAXmitStatus	Transmit Channel A Status
.ChBAudioModeRdg	Channel B Mode Reading
.ChBAuxBitsRdg	Channel B Auxiliary Bits Reading
.ChBCategoryRdg	Channel B Category Reading
.ChBChModeRdg	Channel B Channel Mode Reading
.ChBChNumRdg	Channel B Channel Number Reading
.ChBClockAccuracyRdg	Channel B Clock Accuracy Reading
.ChBCopyrightRdg	Channel B Copyright Reading
.ChBCrcRdg	Channel B CRC Valid Reading
.ChBDestinationRdg	Channel B Destination Data Reading
.ChBEmphRdg	Channel B Emph Reading
.ChBFlag0_5Rdg	Channel B Reliability Flag 0-5
.ChBFlag6_13Rdg	Channel B Reliability Flag 6-13
.ChBFlag14_17Rdg	Channel B Reliability Flag 14-17
.ChBFlag18_21Rdg	Channel B Reliability Flag 18-21
.ChBFreqModeRdg	Channel B Freq Mode Reading
.ChBLocalAddressRdg	Channel B Local Address Reading
.ChBModeRdg	Channel B Mode Reading
.ChBOriginRdg	Channel B Origin Data Reading
.ChBRefSignalRdg	Channel B Ref Signal Reading
.ChBSampleFreqRdg	Channel B Sample Frequency Reading
.ChBSourceNumRdg	Channel B Source Number Reading
.ChBStatusXferToArray	Transfer Channel B Status Bits to String
.ChBStatusXferToString	Transfer Channel B Status Bits to String
.ChBTimeOfDayRdg	Channel B Time Of Day Reading
.ChBUserBitsRdg	Channel B User Bits Reading
.ChBWordLengthRdg	Channel B Word Length Reading
.ChBXmitData	Transmit Channel B Status
.ChBXmitStatus	Transmit Channel B Status
.Cons.AudioMode	Consumer Audio Mode
.Cons.Category	Consumer Category
.Cons.Channels	Consumer 2-Channel or 4-Channel
.Cons.ChNum	Consumer Channel Number

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.Cons.ClockAccuracy	Consumer Clock Accuracy
.Cons.Copyright	Consumer Copyright
.Cons.Emphasis	Consumer Emphasis
.Cons.SampleFreq	Consumer Sample Frequency
.Cons.SourceNum	Consumer Source Number
.Mode	Mode (Consumer or Professional)
.Pro.AudioMode	Professional Audio Mode
.Pro.AuxBits	Professional Aux Bits
.Pro.ChMode	Professional Channel Mode
.Pro.CrcEnable	Professional Crc Bit enable
.Pro.Destination	Professional Destination Data
.Pro.Emphasis	Professional Emphasis
.Pro.Flag0-5	Professional Reliability Flag (0-5)
.Pro.Flag6-13	Professional Reliability Flag (6-13)
.ProFlag14-17	Professional Reliability Flag (14-17)
.Pro.Flag18-21	Professional Reliability Flag (18-21)
.Pro.FreqMode	Professional Frequency Mode
.Pro.LocalAddress	Professional Local Address
.Pro.LocalAddressAuto	Professional Address Auto
.Pro.Origin	Professional Origin Data
.Pro.RefSignal	Professional Reference Signal
.Pro.SampleFreq	Professional Sample Frequency
.Pro.TimeOfDay	Professional Time of Day
.Pro.UserBits	Professional User Bits
.Pro.WordLength	Professional Word Length
.XmitChannel	Transmit Channel

RS-232

AP.CommA *

AP.CommB *

.Break	Sends break signal
.CDHolding	Determines the state of the Carrier Detect line

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.CDTimeout	Determines the time period after the carrier is detected before OnComm event is generated
.CommEvent	Returns commumication event or error
.CommID	Returns communications device handle
.CommPort	Communications port number
.CTSHolding	Determines if Clear To Send data
.CTSTimeout	Determines the time period after the Clear To Send line is low before OnComm event is generated
.DSRHolding	Determines the state of the Data Set Ready line
.DSRTimeout	
.DTREnable	Determines the Data Terminal Ready line state
.Handshaking	Determines hardware handshaking protocol
.InBufferCount	Determines number of characters in receive buffer
.InBufferSize	Determines receive buffer size
.Input	Returns string of characters from receive buffer
.InputLen	Determines number of characters in recieve buffer
.Interval	
.NullDiscard	Determines if null characters are transfered to recieve buffer
.OutBufferCount	Determines number of characters waiting in transmit buffer
.OutBufferSize	Determines the transmit buffer size
.Output	Sends a string of characters to the output buffer
.ParityReplace	Sets replacement character used when parity error occurs
.PortOpen	Determines port status
.Rthreshold	Determines number of characters to

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

	recieve before event is generated
.RTSEnable	Determines the Request To Send line state
.Settings	Determines baud rate, parity, data bit, and stop bit parameters
.Sthreshold	Determines number of characters to transmit before event is generated

Computes

AP.Compute *	
.Avg.Apply	Average Apply NOW
.Avg.Data	Average Data to SAME Data
.Avg.PostSweep	Average Apply after sweep
.Avg.Start	Average Start Value
.Avg.Stop	Average Stop Value
.Center.Apply	Center Apply NOW
.Center.Data	Center Data to SAME Data
.Center.PostSweep	Center Apply after sweep
.Center.Start	Center Start Value
.Center.Stop	Center Stop Value
.Clear.All	Clear all compute settings
.Delta.Apply	Delta Apply NOW
.Delta.Data	Delta Data to SAME Data
.Delta.FileName	Delta File Name
.Delta.PostSweep	Delta Apply after sweep
.Equalize.Apply	Equalize Apply NOW
.Equalize.Data	Equalize Data to SAME Data
.Equalize.FileName	Equalize File Name
.Equalize.PostSweep	Equalize Apply after sweep
.Invert.Apply	Invert Apply NOW
.Invert.Data	Invert Data
.Invert.Horizontal	Invert Horizontal Value
.Invert.PostSweep	Invert Apply after sweep
.Linearity.Apply	Linearity Apply NOW
.Linearity.Data	Linearity Data to SAME Data
.Linearity.PostSweep	Linearity Apply after sweep
.Linearity.Start	Linearity Start Value
.Linearity.Stop	Linearity Stop Value

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.Max.Apply	Max Apply NOW
.Max.Data	Max Data to SAME Data
.Max.PostSweep	Max Apply after sweep
.Max.Start	Max Start Value
.MaxStop	Max Stop Value
.Min.Apply	Min Apply NOW
.Min.Data	Min Data to SAME Data
.Min.Post.Sweep	Min Apply after sweep
.Min.Start	Min Start Value
.Min.Stop	Min Stop Value
.Normalize.Apply	Normalize Apply NOW
.Normalize.Data	Normalize Data to SAME Data
.Normalize.Horizontal	Normalize Horizontal Value
.Normalize.PostSweep	Normalize Apply after sweep
.Normalize.Target	Normalize Target Value
.Sigma.Apply	2-Sigma Apply NOW
.Sigma.Data	2-Sigma Data to SAME Data
.Sigma.PostSweep	2-Sigma Apply after sweep
.Sigma.Start	2-Sigma Start Value
.Sigma.Stop	2-Sigma Stop Value
.Smooth.Apply	Smooth Apply NOW
.Smooth.Auto	Auto smooth data
.Smooth.Data	Smooth Data to SAME Data
.Smooth.Passes	Smooth Passes Value
.Smooth.PostSweep	Smooth Apply after sweep

Data

AP.Data *

.AddRowToEnd	Append data Row to end of data
.ColLimitError	Column Limit Error
.ColLowerLimitError	Column Lower Limit Error
.ColName	Column Name
.ColNumOf	Number of Columns
.ColSize	Column Size
.ColUnit	Column Unit String

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.ColUpperLimitError	Column Upper Limit Error
..DeleteRow	Delete current Row
.Id	Id
.InsertRowAfter	Insert new Row After current row
.InsertRowBefore	Insert new Row Before current row
.LimitError	Limit Error
.LowerLimitError	Lower Limit Error
.OptimizeDisplay	Optimize Display
.UpdateDisplay	Update Display
.UpperLimitError	Upper Limit Error
.Value	Value
.XferToArray	Transfer Data to Array

DCX-127

AP.DCX *

.Ch1DcLevel	DC Out 1
.Ch1DcOutput	DC Out 1 ON/OFF
.Ch2DcLevel	DC Out 2
.Ch2DcOutput	DC Out 2 ON/OFF
.DigInFormat	Digital Input Format
.DigInRdg	Digital Input Reading
.DigInRdgRate	Digital Input Reading Rate
.DigInReady	Digital Input Ready
.DigInScale	Digital Input Scale
.DigInSettling	Digital Input Settling
.DigInTrig	Digital Input Trigger
.DigOut	Digital Output
.DigOutFormat	Digital Output Format
.DigOutScale	Digital Output Scale
.DmmMode	DMM Mode
.DmmOffset	DMM Offset
.DmmRange	DMM Range
.DmmRangeAuto	DMM Range Auto
.DmmRdg	DMM Reading
.DmmRdgRate	DMM Reading Rate
.DmmReady	DMM Ready
.DmmScale	DMM Scale
.DmmSettling	DMM Settling

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.DmmTrig	DMM Trigger
.PortAOutput	Port A Output
.PortBOutput	Port B Output
.PortCOutput	Port C Output

DigitalGenerator

AP.Dgen *

.AmplRatio	Amplitude Ratio
.BurstInterval	Burst Interval
.BurstLevel	Burst Low Level
.BurstOnTime	Burst On Time
.ChAAmpl	Channel A Amplitude
.ChAEqAmpl	Channel A Eq Amplitude
.ChAFreq	Channel A Frequency
.ChAInvert	Channel A Invert
.ChAOutput	Channel A Output ON/OFF
.ChBAmpl	Channel B Amplitude
.ChBEqAmpl	Channel B Eq Amplitude
.ChBFreq	Channel B Frequency
.ChBInvert	Channel B Invert
.ChBOutput	Channel B Output ON/OFF
.ChBTrackA	Channel B Track Channel A
.DitherType	Dither Type
.DualAmplRatio	Dual waveform Amplitude Ratio
.EqCurve	Eq Curve
.Freq	Frequency
.IMCenterFreq	IMD CCIF Center Frequency
.IMFreq	IM Frequency
.IMHighFreq	IMD High Frequency
.Offset	Sine DC Offset
.Output	Output ON/OFF
.Phase	Sine Phase
.RefdBr	Digital Generator dBr
.RefFreq	Digital Generator Freq
.RefVFS	Digital Generator Volts FS
.StepRate	Walking (0) & (1) Rate
.Wfm	Waveform Type
.WfmName	Waveform Name

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

File

AP.File *	
.AppendData	Append file data into memory
.ExportASCIIData	Export data to ASCII file
.ExportGraphic	Export Graphic image
.ImportASCIIData	Import ASCII data into memory
.ImportDOSS1Test	Import System One Test
.OpenData	Load contents of data file into memory
.OpenMacro	Load Macro (OLE) into Macro Editor
.OpenTest	Load Test
.OpenWfm	Load Waveform samples into DSP memory
.SaveAll	Save Test and Macros
.SaveDataAs	Save data in memory to specified file
.SaveMacro	Save Macro (OLE)
.SaveMacroAs	Save Macro to specified file (OLE)
.SaveTest	Save Test
.SaveTestAs	Save Test to specified file
.SaveWfm	Save Waveform
.SaveWfmAs	Save Waveform to specified file

Generator Analog

AP.Gen *	
.Ampl	Amplitude
.AmplRatio	D/A Sine Dual Amplitude Ratio
.BurstInterval	Burst Interval
.BurstLevel	Burst Low Level
.BurstOnTime	Burst On Time
.ChAAmpl	Channel A Amplitude
.ChAEqAmpl	Channel A Eq Amplitude
.ChAFreq	Channel A Frequency
.ChAInvert	Channel A Invert
.ChAOutput	Channel A Output ON/OFF
.ChBAmpl	Channel B Amplitude
.ChBEqAmpl	Channel B Eq Amplitude

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.ChBFreq	Channel B Frequency
.ChBInvert	Channel B Invert
.ChBOutput	Channel B Output ON/OFF
.ChBTrackA	Channel B Track Channel A
.Config	Configure Generator output
.DualAmplRatio	Dual waveform Amplitude Ratio
.EqAmpl	Eq Amplitude
.EqCurve	Eq Curve
.Freq	Frequency
.FreqAccuracy	Frequency Accuracy
.IMAmplRatio	IMD SMPTE Amplitude Ratio
.IMCenterFreq	IMD CCIF Center Frequency
.IMFreq	IMD Frequency
.IMHighFreq	IMD High Frequency
.Impedance	Output Impedance
.Output	Output ON/OFF
.Phase	D/A Sine Phase
.RefdBm	Generator dBm Reference
.RefdBr	Generator dBr Reference
.RefdBrAuto	Auto Set Amplitude dBr Referance
.RefFreq	Generator Frequency Reference
.RefFreqAuto	Auto Set Frequency Referance
.RefWatts	Generator Watts Reference
.Wfm	Waveform
.WfmName	Waveform Name

Graph

AP.Graph *

.Comment	Graph Comment Section Text
.CommentShow	Display Comment area in Graph panel
.CopyToSweepPanel	Transfer Axis Information to Sweep Panel X/Y
.CursorPosition	Cursor location by Horizontal Axis values
.CursorRow	Cursor location by row number
.CursorsOn	Display Cursors
.CursorValue	Cursor location Horizontal Axis

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.OptimizeIndividually	Optimize Graph Axis Individually
.OptimizeLeft	Optimize Graph Left Axis only
.OptimizeRight	Optimize Graph Right Axis only
.OptimizeTogether	Optimize Graph Axis together
.RefDataClear	Clear Reference Data from memory
.RefDataShow	Display Reference Data in graph
.RefDataStore	Update Reference data memory with sweep data

Log

AP.Log *

.AddEntry	Add Entry
.Clear	Clear Log File
.Data	Data
.Enable	Enable
.ErrorMessages	Error Messages
.FileActivity	File Activity
.FileName	File Name
.GraphTitle	Graph Title
.PassFailMessages	Pass Fail Messages
.PrintLogFile	Print attached Log File
.TestName	Test Name
.View	View

Macro

AP.Macro *

.IsRunning	Return Macro Status(OLE)
.Name	Return current Macro File Name

Print

AP.Print *

.Data	Print data in tabular form
.Graph	Print graph
.LoadFromTest	Load graph settings from test
.TrackGraph	Printout track graph settings

Prompt

AP.Prompt *

.FontSize	Prompt font size

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.Hide	Remove prompt from view
.IsUp	Is Displayed?
.Position	Prompt position
.Show	Display prompt
.ShowWithContinue	Display prompt with continue button
.ShowWithContinueAnd_ StopSweep	Display prompt with continue button and stop the running sweep
AP.Prompt.Text	Text displayed in prompt

Regulation

AP.Reg *

.IsRunning	Return Regulation Status
.SourceHigh	Source parameter maximum value
.SourceId	Source parameter
.SourceIteration	Source number of iterations
.SourceLow	Source parameter minimum value
.SourceOperation	Regulation algorithum
.SourceStepSize	Algorithum step size
.Start	Run a single regulation cycle
.StartNoWait	Run a single regulation cycle and continue processing Macro commands
.SweepEnable	Enable regulation cycle durring sweep
.TargetId	Parameter to Regulate
.TargetTolerance	Target tolerance
.TargetToleranceMode	Tolerance mode
.TargetValue	Final regulated value
.Timeout	Regulation process Timeout

System One Digital Input/Output

AP.S1Dio *

.DitherType	Dither Type
.InFormat	Input Format
.OutFormat	Output Format
.Rate	Input / Output Sample Rate
.ReceiveSyncRdg	Receive Sync Reading
.Resolution	Output Resolution
.SerialType	Serial Type

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.TransmitSyncRdg	Transmit Sync Reading

Digital Data Analyzer

AP.S1DSP.Bittest *

.Ampl	Amplitude
.ChADataRdg	Channel A Data Reading
.ChADataReady	Channel A Data Ready
.ChADataTrig	Channel A Data Trigger
.ChAErrRdg	Channel A Error Reading
.ChAErrReady	Channel A Error Ready
.ChAErrTrig	Channel A Error Trigger
.ChAOutput	Channel A Output ON/OFF
.ChBDataRdg	Channel B Data Reading
.ChBDataReady	Channel B Data Ready
.ChBDataTrig	Channel B Data Trigger
.ChBErrRdg	Channel B Error Reading
.ChBErrReady	Channel B Error Ready
.ChBErrTrig	Channel B Error Trigger
.ChBOutput	Channel B Output ON/OFF
.ConstantValue	Value
.DataInvalid	Send Data Invalid Bit
.DisplayError	Error Display
.FreezeOnError	Display Data on Error
.Freq	Frequency
.Output	Output ON/OFF
.RdgRate	Reading Rate
.Wfm	Waveform
.WfmDisplay	Wave Display

Codec Tester

AP.S1DSP.Codec *

.Ampl	Amplitude
.Ch1Rdg	Channel 1 Reading
.Ch1Ready	Channel 1 Ready
.Ch1Source	Channel 1 Source
.Ch1Trig	Channel 1 Trigger
.Ch2Rdg	Channel 2 Reading
.Ch2Ready	Channel 2 Ready

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.Ch2Source	Channel 2 Source
.Ch2Trig	Channel 2 Trigger
.ChAOutput	Channel A Output ON/OFF
.ChBOutput	Channel B Output ON/OFF
.FreqCorrection	Freq Correction
.FreqRes	Freq Resolution
.InputFormat	Input Format
.Mode	Measurement Mode
.Output	Output ON/OFF
.TrigCriteria	Trigger Criteria
.TrigSource	Trigger Delay
.Warnings	Warnings
.WfmName	Waveform name
.Window	FFT Window

MultitoneGenerator/Analyzer

AP.S1DSP.FastTest *

.Ampl	Amplitude
.Ch1Rdg	Channel 1 Reading
.Ch1Ready	Channel 1 Ready
.Ch1Source	Channel 1 Input Source
.Ch1Trig	Channel 1 Trigger
.Ch2Rdg	Channel 2 Reading
.Ch2Ready	Channel 2 Ready
.Ch2Source	Channel 2 Input Source
.Ch2Trig	Channel 2 Trigger
.ChAOutput	Channel A Audio Output ON/OFF
.ChBOutput	Channel B Audio Output ON/OFF
.FftLength	FFT Length
.FreqRes	Frequency Resolution
.InputFormat	Input Format
.Mode	Measurement Mode
.Output	Output ON/OFF
.PhaseDisplay	Channel 2 Phase Display
.TrigSource	Triggering
.WfmName	Waveform name
.Window	Window

Triggered Multitone Tester

AP.S1DSP.FastTrig *

* Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.Ampl	Amplitude
.Ch1Rdg	Channel 1 Reading
.Ch1Ready	Channel 1 Ready
.Ch1Source	Channel 1 Input Source
.Ch1Trig	Channel 1 Trigger
.Ch2Rdg	Channel 2 Reading
.Ch2Ready	Channel 2 Ready
.Ch2Source	Channel 2 Input Source
.Ch2Trig	Channel 2 Trigger
.ChAOutput	Channel A Audio Output ON/OFF
.ChBOutput	Channel B Audio Output ON/OFF
.FreqCorrection	Freq Correction
.FreqRes	Freq Resolution
.InputFormat	Input Format
.Mode	Measurement Mode
.Output	Output ON/OFF
.TrigCriteria	Trigger Criteria
.TrigSource	Trigger Source & Delay
.Warnings	Warnings
.WfmName	Waveform name
.Window	FFT Window

Spectrum Analyzer/ Generator

AP.S1DSP.FFTGen *

.Ampl	Amplitude
.Averages	Number of Averages
.Ch1Rdg	Channel 1 Reading
.Ch1Ready	Channel 1 Ready
.Ch1Source	Channel 1 Input Source
.Ch1Trig	Channel 1 Trigger
.Ch2Rdg	Channel 2 Reading
.Ch2Ready	Channel 2 Ready
.Ch2Source	Channel 2 Input Source
.Ch2Trig	Channel 2 Trigger
.ChAOutput	Channel A Audio Output ON/OFF
.ChBOutput	Channel B Audio Output ON/OFF
.FftLength	FFT Length

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.Freq	Frequency
.InputFormat	Input Format
.Output	Output ON/OFF
.SubtractAve	Subtract Average Value
.TrigSource	Trigger Source
.Wfm	Waveform
.WfmDisplay	Wfm Display Mode
.WfmName	Arbitrary Waveform File Name
.Window	FFT Window

Spectrum Analyzer

AP.S1DSP.FFTSlide *

.Ch1Rdg	Channel 1 Reading
.Ch1Ready	Channel 1 Ready
.Ch1Source	Channel 1 Input Source
.Ch1Trig	Channel 1 Trigger
.Ch2Rdg	Channel 2 Reading
.Ch2Ready	Channel 2 Ready
.Ch2Source	Channel 2 Input Source
.Ch2Trig	Channel 2 Trigger
.FftLength	FFT Length
.InputFormat	Input Format
.PreTrig	Pre-Trigger Time
.StartTime	FFT Start Time
.SubtractAve	Subtract Average Value
.TrigPolarity	Trigger Polarity
.TrigSource	Trigger Source
.WfmDisplay	Wfm Display
.Window	FFT Window

Digital Domain Audio Analyzer

AP.S1DSP.GenAnlr *

.Ampl	Amplitude
.ChAOutput	Channel A Output ON/OFF
.ChBOutput	Channel B Output ON/OFF
.EqAmpl	Eq Amplitude
.EqCurve	Eq Curve
.FilterHP	High Pass Filter Frequency

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.Freq	Frequency
.FreqRdg	Frequency Reading
.FreqReady	Frequency Ready
.FreqSettling	Frequency Settling
.FreqTrig	Frequency Trigger
.FuncBPBRFreq	Band Pass / Band Reject Filter Freq
.FuncBPBRTuning	Band Pass / Band Reject Tuning
.FuncBPHarmonic	Band Pass Harmonic
.FuncInput	Function Meter Input Channel
.FuncMode	Function Meter Mode
.FuncRdg	Function Meter Reading
.FuncReady	Function Meter Ready
.FuncSettling	Function Meter Settling
.FuncTrig	Function Meter Trigger
.LevelRdg	Level Reading
.LevelReady	Level Ready
.LevelSettling	Level Settling
.LevelTrig	Level Trigger
.Output	Output ON/OFF
.RdgRate	Reading Rate
.Wfm	Waveform

Narrow Bandpass Filter

AP.S1DSP.Harmonic *

.AmplRdg	Amplitude Reading
.AmplReady	Amplitude Ready
.AmplSettling	Amplitude Settling
.AmplTrig	Amplitude Trigger
.FilterFreq	Filter Freq
.FilterTuning	Filter Tuning Source
.FilterTuningMode	Filter Tuning Mode
.FilterType	Filter Type
.FreqOffset	Freq Offset
.FreqRdg	Filter Frequency Reading
.FreqReady	Filter Frequency Ready
.FreqSettling	Filter Frequency Settling
.FreqTrig	Filter Frequency Trigger
.RdgRate	Reading Rate

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.Source	Source
.Value	Harmonic

Quasi-Anechoic Acoustical Tester /Generator

AP.S1DSP.Mls *

.Ampl	Amplitude
.Ch1Rdg	Channel 1 Reading
.Ch1Ready	Channel 1 Ready
.Ch1Source	Channel 1 Input Source
.Ch1Trig	Channel 1 Trigger
.Ch2Rdg	Channel 2 Reading
.Ch2Ready	Channel 2 Ready
.Ch2Source	Channel 2 Input Source
.Ch2Trig	Channel 2 Trigger
.ChAOutput	Channel A Output ON/OFF
.ChBOutput	Channel B Output ON/OFF
.InputFormat	Input Format
.Output	Output ON/OFF
.Sequence	MLS Sequence
.TimeDelay	Time Delay
.TimeDisplay	Time Domain Display
.WfmDisplay	Wave Display
.WindowETime	Energy-Time Window
.WindowStart	Time Start Window
.WindowStop	Time Stop Window

System One DSP Program Selection

AP.S1DSP.Program	Load SystemOne DSP Program

System Two Digital Input/Output

AP.S2Dio *

.ChAPeakRdg	Channel A Peak Reading
.ChAPeakReady	Channel A Peak Ready
.ChAPeakTrig	Channel A Peak Trigger
.ChBPeakRdg	Channel B Peak Reading
.ChBPeakReady	Channel B Peak Ready
.ChBPeakTrig	Channel B Peak Trigger
.DelayRdg	Delay Reading

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.DelayReady	Delay Ready
.DelaySettling	Delay Settling
.DelayTrig	Delay Trigger
.FlagChAInvalidRdg	Channel A Invalid Reading
.FlagChBInvalidRdg	Channel B Invalid Reading
.FlagCodingRdg	Coding Reading
.FlagConfidenceRdg	Confidence Reading
.FlagInvalidRdg	Channel A & B Invalid Reading
.FlagLockRdg	Lock Reading
.FlagParityRdg	Parity
.InDeEmp	DeEmphasis Filter
.InFormat	Input Format
.InImpedance	AES Zin
.InJitterBW	Jitter Bandwidth
.InJitterDetector	Jitter Detector
.InJitterMode	Jitter Mode
.InMonitorMode	Peak Monitor Mode
.InResolution	Input Resolution
.InScaleFreq	Scale Frequency
.JitterRdg	Jitter Reading
.JitterReady	Jitter Ready
.JitterSettling	Jitter Settling
.JitterTrig	Jitter Trigger
.OutCableSim	Cable Simulation
.OutCM	Common Mode ON/OFF
.OutCMAmpl	Common Mode Amplitude
.OutCMFreq	Common Mode Frequency
.OutDitherType	Dither Type
.OutFormat	Output Format
.OutJitterAmpl	Jitter Amplitude
.OutJitterEQCurve	Jitter Equalization Curve
.OutJitterFreq	Jitter Frequency
.OutJitterType	Jitter Type
.OutNoise	Noise ON/OFF
.OutNoiseAmpl	Noise Amplitude
.OutPreEmp	PreEmphasis Filter
.OutRate	Output Sample Rate
.OutResolution	Output Resolution

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.OutRiseFall	Fixed Rise/Fall ON/OFF
.OutRiseFallTime	Rise/Fall Time
.OuSendInvalid	Send Invalid
.OutVoltage	Voltage
.RateRdg	Sample Rate Reading
.RateReady	Sample Rate Ready
.RateSettling	Sample Rate Settling
.RateTrig	Sample Rate Trigger
.RefRate	Samle Rate Reference
.VoltageRdg	Voltage Reading
.VoltageReady	Voltage Ready
.VoltageSettling	Voltage Settling
.VoltageTrig	Voltage Trigger

Digital Domain Audio Analyzer

AP.S2DSP.Analyzer

.ChACoupling	Channel A DC Coupled
.ChAFreqRdg	Channel A Frequency Reading
.ChAFreqReady	Channel A Frequency Ready
.ChAFreqSettling	Channel A Frequency Settling
.ChAFreqTrig	Channel A Frequency Trigger
.ChALevelRdg	Channel A Level Reading
.ChALevelReady	Channel A Level Ready
.ChALevelSettling	Channel A Level Settling
.ChALevelTrig	Channel A Level Trigger
.ChARange	Channel A Input Range
.ChARangeAuto	Channel A Range Auto
.ChBCoupling	Channel B DC Coupled
.ChBFreqRdg	Channel B Frequency Reading
.ChBFreqReady	Channel B Frequency Ready
.ChBFreqSettling	Channel B Frequency Settling
.ChBFreqTrig	Channel B Frequency Trigger
.ChBLevelRdg	Channel B Level Reading
.ChBLevelReady	Channel B Level Ready
.ChBLevelSettling	Channel B Level Settling
.ChBLevelTrig	Channel B Level Trigger
.ChBRange	Channel B Input Range
.ChBRangeAuto	Channel B Range Auto

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.FilterHP	High Pass Filter
.FilterWeighting	Weighting Filter
.FuncBPBRFreq	Band Pass / Band Reject Filter Freq
.FuncBPBRTuning	Band Pass / Band Reject Tuning
.FuncBPHarmonic	Band Pass Harmonic
.FuncDetector	Function Meter Detector Type
.FuncFilter	Function Meter Filter
.FuncFilterHP	Function Meter Band Width Limiting HiPass
.FuncFilterLP	Function Meter Band Width Limiting LowPass
.FuncInput	Function Meter Channel
.FuncMode	Function Meter Mode
.FuncRange	Function Meter Range
.FuncRangeAuto	Function Meter Range Auto
.FuncRdg	Function Meter Reading
.FuncReady	Function Meter Ready
.FuncSettling	Function Meter Settling
.FuncTrig	Function Meter Trigger
.InputFormat	Input Format
.RdgRate	Reading Rate
.RdgRateV2	Reading Rate

Digital Data Analyzer

AP.S2DSP.Bittest *

.ChADataRdg	Channel A Data Reading
.ChADataReady	Channel A Data Ready
.ChADataTrig	Channel A Data Trigger
.ChAErrRdg	Channel A Error Reading
.ChAErrReady	Channel A Error Ready
.ChAErrTrig	Channel A Error Trigger
.ChBDataRdg	Channel B Data Reading
.ChBDataReady	Channel B Data Ready
.ChBDataTrig	Channel B Data Trigger
.ChBErrRdg	Channel B Error Reading
.ChBErrReady	Channel B Error Ready
.ChBErrTrig	Channel B Error Trigger
.DisplayError	Error Display

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.FreezeOnError	Display Data on Error
.RdgRate	Reading Rate
.Wfm	Waveform

Multitone Audio Analyzer

AP.S2DSP.FastTest *

.Ch1Rdg	Channel 1 Reading
.Ch1Ready	Channel 1 Ready
.Ch1Source	Channel 1 Input Source
.Ch1Trig	Channel 1 Trigger
.Ch2Rdg	Channel 2 Reading
.Ch2Ready	Channel 2 Ready
.Ch2Source	Channel 2 Input Source
.Ch2Trig	Channel 2 Trigger
.FftLength	FFT Length
.FreqRes	Freq Resolution
.InputFormat	Input Format
.Mode	Measurement Mode
.PhaseDisplay	Ch. 2 Phase Display
.Processing	Processing
.TrigDelay	Trigger Delay
.TrigSource	Triggering

FFT Spectrun Analyzer

AP.S2DSP.Fft *

.Averages	Number of Averages
.AveragesType	Averages Mode
.Ch1Rdg	Channel 1 Reading
.Ch1Ready	Channel 1 Ready
.Ch1Source	Channel 1 Input Source
.Ch1Trig	Channel 1 Trigger
.Ch2Rdg	Channel 2 Reading
.Ch2Ready	Channel 2 Ready
.Ch2Source	Channel 2 Input Source
.Ch2Trig	Channel 2 Trigger
.InputFormat	Input Format
.Length	FFT Length
.PreTrig	FFT Pre-trigger Time

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.StartTime	FFT Start Time
.SubtractAve	Subtract Average DC Value
.SubtractDC	Subtract DC Value
.TrigDelay	Trigger Delay
.TrigPolarity	Trigger Polarity
.TrigSensitivity	Trigger Sensitivity
.TrigSource	Trigger Source
.WfmDisplay	Wfm Display
.Window	FFT Window

Digital Interface Analyzer

AP.S2DSP.Intervu *

.AmplVsTime	Amplitude vs Time display mode
.AudioMonitor	Audio Monitor source selection
.Averages	Number of averages
.JitterDetection	Jitter Detection
.TrigSource	Trigger Source selection
.Window	Window selection

Quasi-Anechoic Acoustical Tester

AP.S2DSP.Mls *

.Ch1Rdg	Channel 1 Reading
.Ch1Ready	Channel 1 Ready
.Ch1Source	Channel 1 Input Source
.Ch1Trig	Channel 1 Trigger
.Ch2Rdg	Channel 2 Reading
.Ch2Ready	Channel 2 Ready
.Ch2Source	Channel 2 Input Source
.Ch2Trig	Channel 2 Trigger
.InputFormat	Input Format
.TimeDelay	Time Delay
.TimeDisplay	Time Domain Display
.TrigSource	Trigger Source
.WfmDisplay	Wave Display
.WindowETime	Energy-Time Window
.WindowStart	Time Start Window
.WindowStop	Time Stop Window

DSP Program Selection and References

* = Note: The AP.S2DSP prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.Program	System Two DSP Program selection
.RefCh1dBr	dBr1 Reference
.RefCh2dBr	dBr2 Reference
.RefFreq	Freq Reference
.RefVFS	Volts/FS Reference

Speaker

AP.Speaker *

.Mode	Stereo/Mono Mode
.Source	Source

Sweep

AP.Sweep *

.Append	Append Data
.CopyData1To2	Copy Data 1 settings to Data 2
.CopyData2To1	Copy Data 2 settings to Data 1
.CreateGraph	Create Graph
.CreateTable	Create Table
.Data1.AutoDiv	Data 1 Divisions Auto
.Data1.Autoscale	Data 1 Autoscale
.Data1.Bottom	Data 1 Graph Bottom
.Data1.Div	Data 1 Divisions
.Data1.Id	Data 1 Readings (ID#)
.Data1.Limits	Data 1 Limits
.Data1.LogLin	Data 1 Log/Lin
.Data1.Top	Data 1 Graph Top
.Data2.AutoDiv	Data 2 Divisions Auto
.Data2.Autoscale	Data 2 Autoscale
.Data2.Bottom	Data 2 Graph Bottom
.Data2.Div	Data 2 Divisions
.Data2.Id	Data 2 Readings (ID#)
.Data2.Limits	Data 2 Limits
.Data2.LogLin	Data 2 Log/Lin

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.Data2.Top	Data 2 Graph Top
.Data3.Id	Data 3 Readings (ID#)
.Data3.Limits	Data 3 Limits
.Data4.Id	Data 4 Readings (ID#)
.Data4.Limits	Data 4 Limits
.Data5.Id	Data 5 Readings (ID#)
.Data5.Limits	Data 5 Limits
.Data6.Id	Data 6 Readings (ID#)
.Data6.Limits	Data 6 Limits
.GraphType	Graph Type
.PreSweepDelay	Pre-Sweep Delay
.Recompare	Re-compare to Limits
.Repeat	Repeat Sweep
.Reprocess	Reprocess Data
.Retransform	Re-Transform FFT
.ReverseChannels	Reverse Channels
.SinglePoint	Single Point Sweep
.Source1.AutoDiv	Source 1 Divisions Auto
.Source1.Div	Source 1 Divisions
.Source1.EndOn	End Sweep On
.Source1.Id	Source 1 Setting (ID#)
.Source1.LogLin	Source 1 Log/Lin
.Source1.MinLevel	Minimum Level
.Source1.MinLevelId	Minimum Level Source (ID#)
.Source1.Multiply	Source 1 Multiply
.Source1.Spacing	Spacing
.Source1.Start	Source 1 Start
.Source1.Steps	Source 1 Steps
.Source1.StepSize	Source 1 Stepsize
.Source1.Stop	Source 1 Stop
.Source1.Table	Source 1 Table
.Source2.Id	Source 2 Setting (ID#)
.Source2.LogLin	Source 2 Log/Lin
.Source2.Multiply	Source 2 Multiply
.Source2.Start	Source 2 Start
.Source2.Steps	Source 2 Steps
.Source2.StepSize	Source 2 Stepsize
.Source2.Stop	Source 2 Stop

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

.Start	Run Sweep (F9)
.StartWithAppend	Run Sweep with Append
.StartWithRepeat	Run Sweep with Repeat
.Stereo	Stereo Sweep
.Timeout	Timeout (Per Step)

Switcher

AP.SWR *

.ChABIn	Channel A+B Input
.ChABInOut	Channel A+B Input/Output
.ChABOut	Channel A+B Output
.ChAIn	Channel A Input
.ChAInOut	Channel A Input/Output
.ChAOut	Channel A Output
.ChBIn	Channel B Input
.ChBInOut	Channel B Input/Output
.ChBOffset	Channel B Offset
.ChBOut	Channel B Output
.Mode	Mode
.OutOffset	Output Offset Out to In

Sync

AP.Sync *

.DelayRdg	Sync Frequency Reading
.DelayReady	Sync Frequency Ready
.DelaySettling	Sync Frequency Settling
.DelayTrig	Sync Frequency Trigger
.Freq	Sync Frequency
.FreqRdg	Sync Frequency Reading
.FreqReady	Sync Frequency Ready
.FreqTrig	Sync Frequency Trigger
.Impedance	Sync Input Z
.OutDelay	Delay Out On/Off
.OutDelayFromRef	Delay Out from Ref Out
.OutOfRangeRdg	Out of Range Reading
.Source	Sync Source
.SourceInput	Sync Source Input
.UnLockedRdg	Unlocked Reading

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

User Notes

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

User Notes

* = Note: The AP.XXXX prefix preceeds each of the following commands. For example AP.Anlr.ChACoupling

Analog Analyzer

AP.Anlr.ChACoupling ❷ Property

Syntax **AP.Anlr.ChACoupling**

Data Type Boolean

True	DC coupled.
False	Not DC coupled.

Description This command sets channel A Input Coupling to DC. This enables System Two to DC couple the input to the A/D converter for improved CMRR at low frequencies and increased low frequency measurement capability. By DC coupling the Analog Analyzer and DSP Audio Analyzer inputs DC Volts can be measured.

See Also AP.Anlr.ChBCoupling

Example

```
Sub Main
   AP.Application.NewTest              'Reset panels
   AP.Application.PanelClose(apbPanelAnlrSmall)
   AP.Application.PanelClose(apbPanelAnalogGenSmall)
   AP.Application.PanelOpen(apbPanelAnlrLarge)
   AP.S2Dsp.Program = 1 'Select DSP Audio Analyzer
   AP.Application.PanelOpen(apbDSPPanelLarge)
   AP.S2Dsp.Analyzer.InputFormat = 1  'Select A/D Input
   AP.Anlr.ChACoupling = True
   AP.Anlr.ChBCoupling = True
   AP.S2Dsp.Analyzer.ChACoupling = True
   AP.S2Dsp.Analyzer.ChBCoupling = True
'Get readings.
   Wait 1
   Reading1 = AP.S2Dsp.Analyzer.ChALevelRdg("V")
   Reading2 = AP.S2Dsp.Analyzer.ChBLevelRdg("V")
   Debug.Print "Channel A DC Level = "; _
      Format(Reading1, "#.0000");" VDC"
   Debug.Print "Channel B DC Level = "; _
      Format(Reading2, "#.0000");" VDC"
End Sub
```

Example Output

```
Channel A DC Level = 9.1081 VDC
Channel B DC Level = .0004 VDC
```

AP.Anlr.ChAFreqRdg ❷ Property

Syntax **AP.Anlr.ChAFreqRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	The following units are available: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.

Description This command returns a settled reading for the channel A Frequency meter and zeros the ready count.

See Also AP.Anlr.ChAFreqReady, AP.Anlr.ChAFreqSettling, AP.Anlr.ChAFreqTrig

Example

```
Sub Main
   AP.Application.NewTest        'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.ChAFreqSettling(.5, .0002, "Hz", 3, .03, 1)
   AP.Anlr.ChAFreqTrig           'Trigger new reading.
   Do
      Ready = AP.Anlr.ChAFreqReady  'Get status.
   Loop Until Ready > 0
   Reading = AP.Anlr.ChAFreqRdg("Hz")  'Get reading.
   Debug.Print "Frequency A = ";Format(Reading, _
      "#.0000");" Hz"
End Sub
```

Example Output

```
Frequency A = 1002.9112 Hz
```

AP.Anlr.ChAFreqReady ❷ Property

Syntax **AP.Anlr.ChAFreqReady**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Frequency A 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 `AP.Anlr.ChAFreqRdg` or `AP.Anlr.ChAFreqTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.Anlr.ChAFreqRdg` command will be guaranteed to return quickly.

See Also `AP.Anlr.FuncInput, AP.Anlr.ChAFreqRdg, AP.Anlr.ChAFreqSettling, AP.Anlr.ChAFreqTrig`

Example See example for `AP.Anlr.ChAFreqRdg`.

AP.Anlr.ChAFreqSettling ❷ Method

Syntax **`AP.Anlr.ChAFreqSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.Anlr.ChAFreqRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.Anlr.FuncInput, AP.Anlr.ChAFreqRdg, AP.Anlr.ChAFreqReady, AP.Anlr.ChAFreqTrig`

Example See example for `AP.Anlr.ChAFreqRdg`.

AP.Anlr.ChAFreqTrig ❷ Method

Syntax `AP.Anlr.ChAFreqTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.Anlr.ChAFreqRdg` command. The reading in progress is aborted.

See Also `AP.Anlr.FuncInput, AP.Anlr.ChAFreqRdg, AP.Anlr.ChAFreqReady, AP.Anlr.ChAFreqSettling`

Example See example for `AP.Anlr.ChAFreqRdg`.

AP.Anlr.ChAImpedance ❶❷ Property

Syntax `AP.Anlr.ChAImpedance`

Data Type Integer

The following list is for System One.

0	150 ohms
1	600 ohms
2	100k ohms

The following list is for System Two.

0	300 ohms
1	600 ohms
2	100k ohms

Description This command selects one of the available termination impedances for the Analyzer channel A input.

See Also `AP.Anlr.ChBImpedance`

❶ **Example**

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.Impedance = 2    'Set generator output Z to _
      600 ohms.
   AP.Gen.Ampl("dBm") = 0
```

```
   AP.Gen.Output = 1
   AP.Anlr.ChARangeAuto = 0 'Set input ranging to fixed.
   AP.Anlr.ChARange("V") = 2.5   'Set input range to _
      2.5 Volts.
   AP.Anlr.ChAInput = 0 'Set anlr input to INPUT(XLR).
   AP.Anlr.ChAImpedance = 1   'Set cha A input Z to _
      600 ohms.
   AP.Anlr.FuncSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.FuncTrig           'Trigger new reading.
   Do
      Ready = AP.Anlr.FuncReady  'Get status.
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg("dBm") 'Get settled _
      reading.
   Debug.Print "Channel A Amplitude =";Format(Reading, _
      "#.0000");" dBm"
   AP.Anlr.ChARangeAuto = 1  'Set input ranging to auto.
End Sub
```

Example Output

```
Channel A Amplitude = 0.3079 dBm
```

AP.Anlr.ChAInput

❶❷ Property

Syntax `AP.Anlr.ChAInput`

Data Type Integer

The following list is for System One.

0	Input
1	GenMon
2	Aux

The following list is for System Two.

0	XLR-Bal
1	BNC-Unbal
2	GenMon

Description This command selects the Analog Analyzer channel A Input.

See Also AP.Anlr.ChBInput

Example See example for AP.Anlr.ChAImpedance.

AP.Anlr.ChALevelRdg ❷ Property

Syntax **AP.Anlr.ChALevelRdg(*unit$*)**

Data Type Variant

Description This command returns the channel A level meter settled reading. For System One the AP.Anlr.FuncInput command must be set to channel A.

Parameters

Name	Description
unit$	The following units are available for System One: V, dBu, dBV, dBr, dBg, dBm, W. The following units are available for System Two: V, dBu, dBV, dBr A, dBr B, dBg A, dBg B, dBm, W.

See Also AP.Anlr.FuncMode, AP.Anlr.FuncInput

Example ❷

```
Sub Main
   System = AP.Application.SysType
   If System = "1" Then Debug.Print "System Two only."
   AP.Application.NewTest
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.ChALevelSettling(1, .000025, "V", 3, .03, 1)
   AP.Anlr.ChALevelTrig          'Trigger new reading.
   Do
      Ready = AP.Anlr.ChALevelReady    'Get status.
      Loop Until Ready > 0
   Reading = AP.Anlr.ChALevelRdg("V")  'Get settled _
      reading.
   Debug.Print "Level A Amplitude = ";Format(Reading, _
      "#.0000");" V"
End Sub
```

Example Output

```
Level A Amplitude = 0.9957 V
```

AP.Anlr.ChALevelReady ❷ Property

Syntax `AP.Anlr.ChALevelReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Level A 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 `AP.Anlr.ChALevelRdg` or `AP.Anlr.ChALevelTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.Anlr.ChALevelRdg` command will be guaranteed to return quickly.

See Also `AP.Anlr.FuncInput, AP.Anlr.ChALevelRdg, AP.Anlr.ChALevelSettling, AP.Anlr.ChALevelTrig`

Example See example for `AP.Anlr.ChALevelRdg`.

AP.Anlr.ChALevelSettling ❷ Method

Syntax **`AP.Anlr.ChALevelSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.Anlr.ChALevelRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.Anlr.FuncMode, AP.Anlr.ChALevelRdg, AP.Anlr.ChALevelReady, AP.Anlr.ChALevelTrig`

Example See example for `AP.Anlr.ChALevelRdg.`

AP.Anlr.ChALevelTrig ❷ Method

Syntax `AP.Anlr.ChALevelTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.Anlr.ChALevelRdg` command. The reading in progress is aborted.

See Also `AP.Anlr.ChALevelRdg, AP.Anlr.ChALevelSettling, AP.Anlr.ChALevelTrig`

Example See example for `AP.Anlr.ChALevelRdg`.

AP.Anlr.ChARange ❶❷ Property

Syntax `AP.Anlr.ChARange(`*`unit$`*`)`

Data Type Double

The following values are the range boundaries for the Volts unit: 160, 80, 40, 20, 10, 5, 2.5, 1.2, .600, .300, .160, .08 (and .04 for System Two).

If an arbitrary value between the range boundaries is entered the next higher range will be selected.

Parameters

Name	Description
unit$	The following units are available: V, dBu, dBV

Description This command sets the `AP.Anlr.ChARange` and returns the nominal full scale of the range in use.

See Also `AP.Anlr.ChARangeAuto`

Example See example for `AP.Anlr.ChAImpedance`.

AP.Anlr.ChARangeAuto ❶❷ Property

Syntax `AP.Anlr.ChARangeAuto`

Data Type Boolean

True	Auto range
False	Fixed range

Description This command sets the Analyzer channel A 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 `AP.Anlr.ChARange`

Example See example for `AP.Anlr.ChAImpedance`.

AP.Anlr.ChBCoupling ❷ Property

Syntax **`AP.Anlr.ChBCoupling`**

Data Type Boolean

True	DC coupled.
False	Not DC coupled.

Description This command sets channel B Input Coupling to DC. This enables System Two to DC couple the input to the A/D converter for improved CMRR at low frequencies and increased low frequency measurement capability. By DC coupling the Analog Analyzer and DSP Audio Analyzer inputs DC Volts can be measured.

See Also `AP.Anlr.ChACoupling`

Example See example for `AP.Anlr.ChACoupling`.

AP.Anlr.ChBFreqRdg ❷ Property

Syntax **`AP.Anlr.ChBFreqRdg(`***`unit$`***`)`**

Data Type Variant

Description This command returns a settled reading for the channel B Frequency meter and zeros the ready count.

Parameters

Part	Description

	unit$	The following units are available: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.

See Also `AP.Anlr.ChAFreqReady, AP.Anlr.FreqATrig, AP.Anlr.ChAFreqSettling`

Example

```
Sub Main
   AP.Application.NewTest        'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChBInput = 1
   AP.Anlr.FuncInput = 1
   AP.Anlr.ChBFreqSettling(.5, .0002, "Hz", 3, .03, 1)
   AP.Anlr.ChBFreqTrig           'Trigger new reading.
   Do
      Ready = AP.Anlr.ChBFreqReady  'Get status.
   Loop Until Ready > 0
   Reading = AP.Anlr.ChBFreqRdg("Hz") 'Get settled _
      reading.
   Debug.Print "Frequency B = ";Format(Reading, _
      "#.0000");" Hz"
End Sub
```

Example Output

```
Frequency B = 999.6856 Hz
```

AP.Anlr.ChBFreqReady ❷ Property

Syntax `AP.Anlr.ChBFreqReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Frequency B 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 `AP.Anlr.ChBFreqRdg` or `AP.Anlr.FreqBTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.Anlr.ChBFreqRdg` command will be guaranteed to return quickly.

See Also `AP.Anlr.FuncInput, AP.Anlr.ChBFreqRdg, AP.Anlr.ChBFreqSettling, AP.Anlr.ChBFreqTrig`

Example See example for `AP.Anlr.ChBFreqRdg`.

AP.Anlr.ChBFreqSettling ❷ Method

Syntax **`AP.Anlr.ChBFreqSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.Anlr.ChBFreqRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.Anlr.FuncInput, AP.Anlr.ChBFreqRdg, AP.Anlr.FreqBTrig, AP.Anlr.FreqBReady`

Example See example for `AP.Anlr.ChBFreqRdg`.

AP.Anlr.ChBFreqTrig ❷ Method

Syntax **`AP.Anlr.ChBFreqTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.Anlr.ChBFreqRdg` command. The reading in progress is aborted.

See Also `AP.Anlr.FuncInput, AP.Anlr.ChBFreqRdg, AP.Anlr.ChBFreqReady, AP.Anlr.ChBFreqSettling`

Example See example for `AP.Anlr.ChBFreqRdg`.

AP.Anlr.ChBImpedance

02 Property

Syntax `AP.Anlr.ChBImpedance`

Data Type Integer

The following list is for System One.

0	150 ohms
1	600 ohms
2	100k ohms

The following list is for System Two.

0	300 ohms
1	600 ohms
2	100k ohms

Description This command selects one of the available termination impedances for the Analyzer channel B Input.

See Also `AP.Anlr.ChAImpedance`

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.Impedance = 2    'Set generator output Z to _
      600 ohms.
   AP.Gen.Ampl("dBm") = 0
   AP.Gen.Output = 1
   AP.Anlr.ChBRangeAuto = 0 'Set input ranging to fixed.
   AP.Anlr.ChBRange("V") = 2.5   'Set input range to _
      2.5 Volts.
   AP.Anlr.ChBInput = 0 'Set anlr input to INPUT(XLR).
   AP.Anlr.ChBImpedance = 1   'Set Cha A input Z to _
      600 ohms.
   AP.Anlr.FuncInput = 1   'Set Function Meter Cha to B.
   AP.Anlr.FuncSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.FuncTrig           'Trigger new reading.
   Do
      Ready = AP.Anlr.FuncReady  'Get status.
   Loop Until Ready > 0
```

```
    Reading = AP.Anlr.FuncRdg("dBm")'Get settled _
       reading.
    Debug.Print "Channel B Amplitude = ";Format _
       (Reading, "#.0000");" dBm"
    AP.Anlr.ChBRangeAuto = 1 'Set input ranging to auto.
End Sub
```

Example Output

```
Channel B Amplitude = -103.7187 dBm
```

AP.Anlr.ChBInput

❶❷ Property

Syntax **AP.Anlr.ChBInput**

Data Type Integer

The following list is for System One.

0	Input
1	GenMon

The following list is for System Two.

0	XLR-Bal
1	BNC-Unbal
2	GenMon

Description This command selects the Analog Analyzer channel B Input.

See Also `AP.Anlr.ChAInput`

Example See example for `AP.Anlr.ChBImpedance`.

AP.Anlr.ChBLevelRdg

❷ Property

Syntax **AP.Anlr.ChBLevelRdg(*unit$*)**

Data Type Variant

Parameters

Name	Description

unit$	The following units are available for System One: V, dBu, dBV, dBr, dBg, dBm, W. The following units are available for System Two: V, dBu, dBV, dBr A, dBr B, dBg A, dBg B, dBm, W.

Description This command returns the channel B Level meter settled reading. For System One the `AP.Anlr.FuncInput` command must be set to channel B.

See Also `AP.Anlr.FuncMode, AP.Anlr.FuncInput, AP.Anlr.ChBLevelReady, AP.Anlr.ChBLevelSettling, AP.Anlr.ChBLevelTrig`

Example ❷

```
Sub Main
   System = AP.Application.SysType
   If System = "1" Then Debug.Print "System Two only."
   AP.Application.NewTest
   AP.Gen.Output = 1
   AP.Anlr.ChBInput = 1
   AP.Anlr.ChBLevelSettling(1, .000025, "V", 3, .03, 1)
   AP.Anlr.ChBLevelTrig          'Trigger new reading.
   Do
      Ready = AP.Anlr.ChBLevelReady    'Get status.
      Loop Until Ready > 0
   Reading = AP.Anlr.ChBLevelRdg("V")  'Get settled _
      reading.
   Debug.Print "Level B Amplitude = ";Format(Reading, _
      "#.0000");" V"
End Sub
```

Example Output

```
Level B Amplitude = 0.9973 V
```

AP.Anlr.ChBLevelReady ❷ Property

Syntax `AP.Anlr.ChBLevelReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Level B 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 `AP.Anlr.ChBLevelRdg` or `AP.Anlr.ChBLevelTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.Anlr.ChBLevelRdg` command will be guaranteed to return quickly.

See Also `AP.Anlr.FuncInput`

Example See example for `AP.Anlr.ChBLevelRdg.`

AP.Anlr.ChBLevelSettling ❷ Method

Syntax **`AP.Anlr.ChBLevelSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.Anlr.ChBLevelRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.Anlr.FuncMode, AP.Anlr.ChBLevelRdg, AP.Anlr.ChBLevelReady, AP.Anlr.ChBLevelTrig`

Example See example for `AP.Anlr.ChBLevelRdg.`

AP.Anlr.ChBLevelTrig ❷ Method

Syntax **`AP.Anlr.ChBLevelTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.Anlr.ChBLevelRdg` command. The reading in progress is aborted.

See Also AP.Anlr.FuncInput

Example See example for AP.Anlr.ChBLevelRdg.

AP.Anlr.ChBRange

❶❷ Property

Syntax **AP.Anlr.ChBRange(*unit$*)**

Data Type Double

The following values are the range boundaries for the Volts unit: 160, 80, 40, 20, 10, 5, 2.5, 1.2, .600, .300, .160, .08 (and .04 for System Two).

If an arbitrary value between the range boundaries is entered the next higher range will be selected.

Parameters

Name	Description
unit$	The following units are available: V, dBu, dBV

Description This command sets the AP.Anlr.ChBRange and returns the nominal full scale of the range in use.

See Also AP.Anlr.ChBRangeAuto

Example See example for AP.Anlr.ChBImpedance.

AP.Anlr.ChBRangeAuto

❶❷ Property

Syntax **AP.Anlr.ChBRangeAuto**

Data Type Boolean

True	Auto range
False	Fixed range

Description This command sets the Analyzer channel B 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 AP.Anlr.ChBRange

Example See example for AP.Anlr.ChBImpedance.

AP.Anlr.FreqRdg

Property

Syntax **AP.Anlr.FreqRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	The following units are available: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.

Description This command returns a settled reading for the Frequency meter and zeros the ready count.

See Also AP.Anlr.FreqReady, AP.Anlr.FreqSettling, AP.Anlr.FreqTrig

Example

```
Sub Main
   AP.Application.NewTest        'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.FuncInput = 1
   AP.Anlr.FreqSettling(.5, .0002, "Hz", 3, .03, 1)
   AP.Anlr.FreqTrig              'Trigger new reading.
   Do
      Ready = AP.Anlr.FreqReady     'Get status.
   Loop Until Ready > 0
   Reading = AP.Anlr.FreqRdg("Hz") 'Get settled reading.
   Debug.Print "Frequency B = ";Format(Reading, _
      "#.0000");" Hz"
End Sub
```

Example Output

```
Frequency B = 1002.9112 Hz
```

AP.Anlr.FreqReady

Property

Syntax **AP.Anlr.FreqReady**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the 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 `AP.Anlr.FreqRdg` or `AP.Anlr.FreqTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.Anlr.FreqRdg` command will be guaranteed to return quickly.

See Also `AP.Anlr.FuncInput, AP.Anlr.FreqRdg, AP.Anlr.FreqSettling, AP.Anlr.FreqTrig`

Example See example for `AP.Anlr.FreqRdg`.

AP.Anlr.FreqSettling

Method

Syntax **`AP.Anlr.FreqSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.Anlr.FreqRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.Anlr.FuncInput, AP.Anlr.FreqRdg, AP.Anlr.FreqReady, AP.Anlr.FreqTrig`

Example See example for `AP.Anlr.FreqRdg`.

AP.Anlr.FreqTrig

Method

Syntax **`AP.Anlr.FreqTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.Anlr.FreqRdg` command. The reading in progress is aborted.

See Also `AP.Anlr.FuncInput, AP.Anlr.FreqRdg, AP.Anlr.FreqReady, AP.Anlr.FreqSettling`

Example See example for `AP.Anlr.FreqRdg`.

AP.Anlr.FuncBPBRFreq ❶❷ Property

Syntax **`AP.Anlr.FuncBPBRFreq(`*`unit$`*`)`**

Data Type Double — Any frequency value between 10 Hz to 204 kHz.

Parameters

Name	Description
`unit$`	The following units are available: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.

Description This command sets the bandpass/bandreject filter frequency.

See Also `AP.Anlr.FuncBPBRTuning, AP.Anlr.FuncMode`

Example

```
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest        'Reset panels
   AP.Gen.Output = 1
   Frequency = AP.Gen.Freq("Hz")
   AP.Anlr.FuncMode = 1    'Set Anlr mode to Bandpass
   If System = "1" Then
      AP.Anlr.ChAInput = 1
      AP.Anlr.FuncBPBRTuning = 3 'Set Tuning to Fixed _
         for System One
   Else
      AP.Anlr.ChAInput = 2
      AP.Anlr.FuncBPBRTuning = 4 'Set Tuning to Fixed _
         for System Two
   End If
   For Harmonic = 2 To 5 Step 1
      AP.Anlr.FuncBPBRFreq("Hz") = Frequency * Harmonic
      AP.Anlr.FuncSettling(1, .000002, "V", 4, .05, 1)
      AP.Anlr.FuncTrig
      Do
         Ready = AP.Anlr.FuncReady
      Loop Until Ready > 0
```

```
        Reading = AP.Anlr.FuncRdg("dBV")
        Debug.Print "Harmonic Amplitude at ";Frequency * _
           Harmonic;" Hz = ";Format(Reading, "#.0000");" _
           dBV"
    Next Harmonic
End Sub
```

Example Output

```
Harmonic Amplitude at 2000 Hz = -33.0766 dBV
Harmonic Amplitude at 3000 Hz = -43.0576 dBV
Harmonic Amplitude at 4000 Hz = -48.9417 dBV
Harmonic Amplitude at 5000 Hz = -53.2414 dBV
```

Comment

This example Macro sets the Analog Generator to 1kHz and sweeps the Bandpass filter through the 2nd to 5th harmonics. A settled reading is taken at each harmonic frequency and displayed on the Debug Immediate Tab.

AP.Anlr.FuncBPBRTuning

❶❷ Property

Syntax `AP.Anlr.FuncBPBRTuning`

Data Type Integer

The following list is for System One.

0	Counter tuned
1	Sweep track
2	Analog Generator track
3	Fixed frequency

The following list is for System Two.

0	Counter tuned
1	Sweep track
2	Analog Generator track
3	Digital Generator track
4	Fixed frequency

Description This command sets the Bandpass Bandreject filter tuning source.

See Also `AP.Anlr.FuncBPBRFreq`

Example See example for `AP.Anlr.FuncBPBRFreq`.

AP.Anlr.FuncDetector ❶❷ Property

Syntax **`AP.Anlr.FuncDetector`**

Data Type Integer

0	RMS
1	Average
2	Peak
3	Qpeak
4	Peak-Equivalent-Sine

Description This command selects the Detector Type for Function meter.

See Also `AP.Anlr.FuncInput, AP.Anlr.RdgRate, AP.Anlr.FuncMode, AP.Anlr.FuncRange, AP.Anlr.FuncRangeAuto`

Example See example for `AP.Anlr.FuncInput`.

AP.Anlr.FuncFilter ❶❷ Property

Syntax **`AP.Anlr.FuncFilter`**

Data Type Integer

The following list is for System One.

0	no weighting
1	weighting slot #1
2	weighting slot #2
3	weighting slot #3
4	weighting slot #4
5	weighting slot #5
6	external weighting filter

7	weighting slot #1 with a gain of 2.092 (for CCIR filter)
8	weighting slot #1 with a gain of 4.0 (for CCIR filter)
	Additionally available with the version “A” hardware:

The following list is for System Two.

0	no weighting
1	weighting slot #1
2	weighting slot #2
3	weighting slot #3
4	weighting slot #4
5	weighting slot #5
6	external weighting filter
7	weighting slot #1 with a gain of 2.092 (for CCIR filter)
8	weighting slot #1 with a gain of 4.0 (for CCIR filter)
	Additionally available with the version “A” hardware:

Description

This command selects one or none of the available weighting filters.

The weighting filters are optional filters that plug into the analyzer (internally), so the exact function of each of the selections here is dependent on the filters installed.

Any other setting attempted results in no weighting being selected (weighting 0).

SPECIAL NOTES FOR SLOT #1:

Normally weighting slot #1 has special gain selections for use with the “CCIR” weighting filter.

Because of the design of that filter, the reading will be corrected internally in the software whenever 7 or 8 is selected. The selection for CCIR-468 compliance is 8 with a correction gain of 4.0 (causing the CCIR filter to cross 0 dB at 1 kHz.).

As a second choice for CCIR, selecting 7 causes slot #1 to be used with a correction factor of 2.09 (causing the “CCIR” filter to cross 0 dB at 2 kHz for DOLBY-ARM measurements).

Weighting slot #2 is normally reserved for “A” weighting.

See Also

`AP.Anlr.FuncFilterHP, AP.Anlr.FuncFilterLP`

Example

```
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = 1
   If System = "1" Then
      AP.Anlr.ChAInput = 1
   Else
      AP.Anlr.ChAInput = 2
   End If
   AP.Sweep.Data1.Id = 5906 'Set Sweep Data 1 to _
      "Anlr.Ampl"
   AP.Anlr.FuncFilter = 3  'Set filter to "A" Weighting
   AP.Sweep.Start
End Sub
```

Comment

The example program produces a graph that displays the frequency response for the "A" Weighting filter.

AP.Anlr.FuncFilterHP

❶❷ Property

Syntax `AP.Anlr.FuncFilterHP`

Data Type Integer

0	<10 Hz
1	22 Hz
2	100 Hz
3	400 Hz

Description

This command selects the value of High Pass filter in the function meter circuit.

See Also `AP.Anlr.FuncFilterLowPass`

Example

```
Sub Main
   AP.Application.NewTest        'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.FuncFilterHP = 2 'Set High Pass filter to _
      100Hz.
```

```
   A = AP.Anlr.FuncFilterHP 'Return High Pass filter _
      setting value.
   AP.Sweep.Data1.Id = 5906
   AP.Sweep.Start
End Sub
```

Comment The example program produces a graph that displays the frequency response for the 100Hz High Pass filter.

AP.Anlr.FuncFilterLP ❶❷ Property

Syntax `AP.Anlr.FuncFilterLP`

Data Type Integer

0	22 kHz
1	30 kHz
2	80 kHz
3	>500 kHz

Description This command selects the value of Low Pass filter in the function meter circuit.

See Also `AP.Anlr.FuncFilterHP`

Example

```
Sub Main
   AP.Application.NewTest        'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.FuncFilterLP = 1   'Set Low Pass filter to _
      30kHz.
   A = AP.Anlr.FuncFilterLP   'Return Low Pass filter _
      setting value.
   AP.Sweep.Data1.Id = 5906
   AP.Sweep.Source1.Start("Hz") = 100000
   AP.Sweep.Start
End Sub
```

Comment The example program produces a graph that displays the frequency response for the 30kHz Low Pass filter.

AP.Anlr.FuncInput

❶❷ Property

Syntax `AP.Anlr.FuncInput`

Data Type Integer

0	Channel A
1	Channel B

Description This command selects channel A or channel B to be used for measurements with the Function meter.

See Also `AP.Anlr.RdgRate, AP.Anlr.FuncDetector, AP.Anlr.FuncMode, AP.Anlr.FuncRange, AP.Anlr.FuncRangeBAuto`

Example

```
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest        'Reset panels
   AP.Gen.Output = 1
   If System = "1" Then
      AP.Gen.Ampl("Vrms") = .01
      AP.Anlr.ChAInput = 1
   Else
      AP.Gen.ChAAmpl("Vrms") = .01
      AP.Anlr.ChAInput = 2
   End If
   AP.Anlr.FuncMode = 0 'Set Function Meter mode to _
      amplitude.
   AP.Anlr.FuncInput = 0      'Set Function Meter _
      channel to A.
   AP.Anlr.FuncRangeAuto = 0  'Set Function Meter _
      range to fixed.
   AP.Anlr.FuncRange("X/Y") = 4  'Set Function Meter _
      range to 4.0 X/Y.
   AP.Anlr.FuncDetector = 1   'Set Function Meter _
      Average.
   AP.Anlr.RdgRate = 1        'Set reading rate to _
      4/Sec.
   AP.Anlr.FuncSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.FuncTrig           'Trigger new reading.
   Do
```

```
      Ready = AP.Anlr.FuncReady  'Get status.
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg("dBV")'Get settled reading.
   Debug.Print "Channel A Averaged Amplitude = _
      ";Format(Reading, "#.0000");" dBV"
End Sub
```

Example Output Channel A Averaged Amplitude = -40.0410 dBV

AP.Anlr.FuncMode ❶❷ Property

Syntax `AP.Anlr.FuncMode`

Data Type Integer

The following list is for System One.

0	Amplitude
1	Bandpass
2	Bandreject
3	THD+N Amplitude
4	THD+N Ratio
5	SMPTE
6	CCIF
7	DIM
8	Wow & Flutter
9	2-Ch. Amplitude
10	2-Ch. Ratio
11	2-Ch. BP Amplitude
12	Crosstalk
13	DFD

The following list is for SystemTwo.

0	Amplitude
1	Bandpass
2	Bandreject
3	THD+N Amplitude

4	THD+N Ratio
5	SMPTE
6	CCIF
7	DIM
8	Wow & Flutter
9	2-Ch. Ratio
10	Crosstalk
11	DFD

Description This command selects the analysis mode of the Analyzer Function meter.

The measurement is taken from the selected channel, using the selected mode, using the unit specified by that mode.

If a reading is not ready when this function is called, it will wait for a reading to become available. Any particular reading will be returned only once.

See Also `AP.Anlr.FuncInput, AP.Anlr.FuncReady, AP.Anlr.FuncSettling, AP.Anlr.FuncTrig, AP.Anlr.RdgRate, AP.Anlr.FuncRange, AP.Anlr.FuncRangeAuto`

Example See example for `AP.Anlr.FuncInput`.

AP.Anlr.FuncRange ❶❷ Property

Syntax **`AP.Anlr.FuncRange(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
unit$	The following units are available: X/Y, dB.

Description This command sets the Analyzer Function meter Range.

The following table shows the gains (X/Y) and (dB) available and to what full-scale ranges they correspond for the various measurement modes of the analyzer:

X/Y	dB	Ampl(S1)	Ampl(S2)	THD & DIM	CCIF/SMPTE
1	0	80 mV	40 mV	100%	25%
4	12.041	20 mV	10 mV	25%	6%
16	24.082	5 mV	2.5 mV	6%	1.6%
64	36.124	1.2 mV	600 uV	1.6%	0.4%
256	48.165	300 uV	150 uV	0.4%	0.1%
1024	60.206	*75 uV	*40 uV	0.1%	0.025%

* a gain of 1024 is valid only for Bandpass, Bandreject, THD+N, and Crosstalk measurements.

Note that for the amplitude ranges, the `AP.Anlr.ChARange` and `AP.Anlr.ChBRange` must be set to the 80 mV range (System One) or 40 mV range (System Two) before these ranges are valid. Likewise, the gain here should be set to 1 or Auto (See command : `AP.Anlr.FuncRangeAuto`) if the input range is set to anything other than 80 mV (System One) or 40 mV (System Two). While the Function meter ranges may be set independently of the input range settings, specified operation cannot be guaranteed if these cautions are not observed.

This range must be reprogrammed if the measurement mode of the Analyzer Function meter is changed (See `AP.Anlr.FuncMode`). Otherwise, the resulting range is not determinate.

A common use of this command is to set the Function meter Range by obtaining the gain reading while in auto range and then set the gain to the determined range. This keeps the Function meter Range from changing during an acquisition.

See Also `AP.Anlr.FuncInput, AP.Anlr.RdgRate, AP.Anlr.WFDetector, AP.Anlr.FuncMode, AP.Anlr.FuncRangeAuto`

Example See example for `AP.Anlr.FuncInput`.

AP.Anlr.FuncRangeAuto

❶❷ Property

Syntax **`AP.Anlr.FuncRangeAuto`**

Data Type Boolean

True	Auto range.
False	Fixed range.

Description This command sets the Function meter to Auto or Fixed Range.

See Also `AP.Anlr.FuncInput, AP.Anlr.RdgRate, AP.Anlr.FuncMode, AP.Anlr.FuncRange`

Example See example for `AP.Anlr.FuncInput`.

AP.Anlr.FuncRdg

❶❷ Property

Syntax **`AP.Anlr.FuncRdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description
unit$	The following units (V, dBu, dBV, dBr, dBg, dBm, W for System One) or (V, dBu, dBV, dBr A, dBr B, dBg A, dBg B, dBm, W for System Two) are available for the following Function meter Modes: Amplitude, Bandpass, Bandreject, THD+N Amplitude, (2-Channel Amplitude, and 2-Channel Band Pass Amplitude for System One only). The following units (%, dB, PPM, X/Y) are available for the following Function meter Modes: THD+N Ratio, SMPTE, CCIF, DIM, Wow & Flutter, 2-Channel Ratio, and Crosstalk.

Description This command returns a reading from the Function meter and zeros the ready count.

See Also `AP.Anlr.FuncInput, AP.Anlr.FuncMode, AP.Anlr.FuncReady, AP.Anlr.FuncSettling, AP.Anlr.FuncTrig`

Example

```
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest        'Reset panels
   AP.Gen.Freq("Hz") = 3150
   If System = "1" Then             'System One
```

```
      AP.Gen.Ampl("dBu") = 0.0
      AP.Anlr.ChAInput = 1        'GenMon
   Else                           'System Two
      AP.Gen.ChAAmpl("dBu") = 0.0
      AP.Anlr.ChAInput = 2     'GenMon input
   End If
   AP.Gen.Output = 1
   AP.Anlr.FuncMode = 8        'Select W&F mode
   AP.Anlr.FuncInput = 0       'Select Channel A input
   AP.Anlr.WFDetector = 1      'Set W&F detector to JIS
   AP.Anlr.WFFilter = 1 'Set W&F Filter to UnWeighted
   AP.Anlr.FuncSettling(5, .0002, "%", 3, .05, 1)
   AP.Anlr.FuncTrig            'Trigger new reading
   Do
      Ready = AP.Anlr.FuncReady  'Get status
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg("%") 'Get settled reading
   Debug.Print "Wow & Flutter = ";Format(Reading, _
      "#.00000");" %"
End Sub
```

Example Output Wow & Flutter = .05305 %

AP.Anlr.FuncReady

02 Property

Syntax `AP.Anlr.FuncReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the 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 `AP.Anlr.FuncRdg` or `AP.Anlr.FuncTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.Anlr.FuncRdg` command will be guaranteed to return quickly.

See Also `AP.Anlr.FuncInput, AP.Anlr.FuncRdg, AP.Anlr.FuncSettling, AP.Anlr.FuncTrig`

Example See example for `AP.Anlr.FuncRdg`.

AP.Anlr.FuncSettling

01 02 Method

Syntax **`AP.Anlr.FuncSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.Anlr.FuncRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.Anlr.FuncInput, AP.Anlr.FuncRdg, AP.Anlr.FuncReady, AP.Anlr.FuncTrig`

Example See example for `AP.Anlr.FuncRdg`.

AP.Anlr.FuncTrig

01 02 Method

Syntax **`AP.Anlr.FuncTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.Anlr.FuncRdg` command. The reading in progress is aborted.

See Also `AP.Anlr.FuncInput, AP.Anlr.FuncRdg, AP.Anlr.FuncReady, AP.Anlr.FuncSettling`

Example See example for `AP.Anlr.FuncRdg`.

AP.Anlr.LevelRdg

Property

Syntax **AP.Anlr.LevelRdg(*unit$*)**

Data Type Variant

Parameters

Name	Description
unit$	The following units are available: V, dBu, dBV, dBr, dBg, dBm, W.

Description This command returns the Level meter settled reading.

See Also AP.Anlr.FuncMode, AP.Anlr.FuncInput, AP.Anlr.LevelReady, AP.Anlr.LevelSettling, AP.Anlr.LevelTrig

Example

```
Sub Main
   AP.Application.NewTest        'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChBInput = 1
   AP.Anlr.FuncInput = 1
   AP.Anlr.LevelSettling(1, .000025, "V", 3, .03, 1)
   AP.Anlr.LevelTrig             'Trigger new reading.
   Do
      Ready = AP.Anlr.LevelReady 'Get status.
   Loop Until Ready > 0
   Reading = AP.Anlr.LevelRdg("V") 'Get settled reading.
   Debug.Print "Level B amplitude = ";Format(Reading, _
      "#.0000");" V"
End Sub
```

Example Output

```
Level B amplitude = 0.9973 V
```

AP.Anlr.LevelReady

Property

Syntax **AP.Anlr.LevelReady**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Level 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 `AP.Anlr.LevelRdg` or `AP.Anlr.LevelTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.Anlr.LevelRdg` command will be guaranteed to return quickly.

See Also `AP.Anlr.FuncInput`

Example See example for `AP.Anlr.LevelRdg.`

AP.Anlr.LevelSettling

Method

Syntax **`AP.Anlr.LevelSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.Anlr.LevelRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.Anlr.FuncMode, AP.Anlr.LevelRdg, AP.Anlr.LevelReady, AP.Anlr.LevelTrig`

Example See example for `AP.Anlr.ChBLevelRdg.`

AP.Anlr.LevelTrig

Method

Syntax **`AP.Anlr.LevelTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.Anlr.LevelRdg` command. The reading in progress is aborted.

See Also `AP.Anlr.FuncInput`

Example See example for `AP.Anlr.LevelRdg.`

AP.Anlr.PhaseMode

12 Property

Syntax `AP.Anlr.PhaseMode`

Data Type Integer

0	Auto
1	-180 +180 deg
2	0 +360 deg
3	-90 +270 deg

Description This function sets the Analog Analyzer Phase measurement range.

Example See example for `AP.Anlr.PhaseRdg`.

AP.Anlr.PhaseRdg

12 Property

Syntax `AP.Anlr.PhaseRdg(`*unit$*`)`

Data Type Variant

Parameters

Name	Description
unit$	The following units are available: deg.

Description This command returns the Analog Analyzer Phase meter settled reading.

Example

```
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest        'Reset panels
   AP.Gen.ChBInvert = 1
   AP.Gen.Output = 1
   If System = "1" Then
      AP.Anlr.ChAInput = 1
      AP.Anlr.ChBInput = 1
   Else
      AP.Anlr.ChAInput = 2
      AP.Anlr.ChBInput = 2
   End If
```

```
    AP.Anlr.PhaseMode = 0
    AP.Anlr.PhaseSettling(0, .5, "deg", 3, .03, 1)
    AP.Anlr.PhaseTrig              'Trigger new reading.
    Do
       Ready = AP.Anlr.PhaseReady 'Get status.
    Loop Until Ready > 0
    Reading = AP.Anlr.PhaseRdg("deg")'Get settled reading
    Debug.Print "Phase A to B = ";Format(Reading, _
       "#.0000");" deg"
End Sub
```

Example Output

```
Phase A to B = 179.9375 deg
```

AP.Anlr.PhaseReady

❶❷ Property

Syntax `AP.Anlr.PhaseReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Phase 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 `AP.Anlr.PhaseRdg` or `AP.Anlr.PhaseTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.Anlr.PhaseRdg` command will be guaranteed to return quickly.

See Also `AP.Anlr.PhaseMode, AP.Anlr.PhaseRdg, AP.Anlr.PhaseSettling, AP.Anlr.PhaseTrig`

Example See example for `AP.Anlr.PhaseRdg`.

AP.Anlr.PhaseSettling ❶❷ Method

Syntax **AP.Anlr.PhaseSettling(***0, floor#, floorunit$, points%, delay#, algorithm%***)**

Description This command sets the settling parameters for the AP.Anlr.PhaseRdg command. Note that this command doesn't require a tolerance setting as in all other settling commands. Enter a 0 (Zero) as shown above for the first parameter as a place holder for the tolerance setting.

See Appendix C for Settling Algorithm and command part descriptions.

See Also AP.Anlr.PhaseMode, AP.Anlr.PhaseRdg, AP.Anlr.PhaseReady, AP.Anlr.PhaseTrig

Example See example for AP.Anlr.PhaseRdg.

AP.Anlr.PhaseTrig ❶❷ Method

Syntax **AP.Anlr.PhaseTrig**

Description Causes a restart of the reading cycle and zeros the ready count for the AP.Anlr.PhaseRdg command. The reading in progress is aborted.

See Also AP.Anlr.PhaseMode, AP.Anlr.PhaseRdg, AP.Anlr.PhaseReady, AP.Anlr.PhaseSettling

Example See example for AP.Anlr.PhaseRdg.

AP.Anlr.RdgRate ❶❷ Property

Syntax **AP.Anlr.RdgRate**

Data Type Integer

The following list is for System One.

0 Auto reading rate. The reading rate is automatically selected based on the measured frequency.

1	4/Sec fixed rate.
2	8/Sec fixed rate.
3	16/Sec fixed rate.
4	32/Sec fixed rate.

The following list is for System Two.

0	Auto reading rate. The reading rate is automatically selected based on the measured frequency.
1	4/Sec fixed rate.
2	8/Sec fixed rate.
3	16/Sec fixed rate.
4	32/Sec fixed rate.
5	64/Sec fixed rate.
6	128/Sec fixed rate.
7	Auto-Fast.
8	Auto-Precise.

Description

This command sets the detector averaging time for the RMS and Average detectors.

These functions have no effect on the Peak and Qpeak detectors.

There is an inherent relationship between the detector averaging time and the lowest frequency component of the measured signal. The combinations of detector time constant and reading rate will affect both low frequency accuracy and digit stability.

For most applications, detector time constants should be ganged with reading rate, where the slowest time constant (range 1) is used for 4 readings/second, and the fastest (range 4) for 30 readings/sec.

See Also

`AP.Anlr.FuncInput, AP.Anlr.FuncDetector, AP.Anlr.FuncMode, AP.Anlr.FuncRange, AP.Anlr.FuncRangeAuto`

Example

See example for `AP.Anlr.FuncInput`.

AP.Anlr.RefChAdBr ❷ Property

Syntax **AP.Anlr.RefChAdBr(*unit$*)**

Data Type Double Reference value.

Parameters

Name	Description
unit$	The following units are available: V, dBV, dBu

Description This command sets the zero dB value of Analog Analyzer dBr A reference.

Example

```
Sub Main
   AP.Application.NewTest           'Reset panels
   AP.Gen.ChAAmpl("Vrms") = 1
   AP.Gen.ChBTrackA = 1
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 2
   AP.Anlr.ChBInput = 2
   AP.Anlr.RefChAdBr("V") = 1
   AP.Anlr.RefChBdBr("V") = 1
   ReferenceA = AP.Anlr.RefChAdBr("V")
   ReferenceB = AP.Anlr.RefChBdBr("V")
   AP.Anlr.ChALevelSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.ChBLevelSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.ChALevelTrig 'Trigger new reading.
   AP.Anlr.ChBLevelTrig 'Trigger new reading.
   Do
      ReadyA = AP.Anlr.ChALevelReady   'Get status.
      ReadyB = AP.Anlr.ChBLevelReady   'Get status.
   Loop Until (ReadyA > 0 And ReadyB > 0)
   ReadingA = AP.Anlr.ChALevelRdg("dBr A")'Get settled _
      reading.
   ReadingB = AP.Anlr.ChBLevelRdg("dBr B")'Get settled _
      reading.
   Debug.Print "Channel A Gain = ";Format(ReadingA, _
      "#.0000");" dBr relative to";ReferenceA;" Volts"
   Debug.Print "Channel B Gain = ";Format(ReadingB, _
      "#.0000");" dBr relative to";ReferenceB;" Volts"
End Sub
```

Example Output

```
Channel A Gain = -.0296 dBr relative to 1 Volts
```

```
Channel B Gain = -.0073 dBr relative to 1 Volts
```


AP.Anlr.RefChBdBr ❷ Property

Syntax **AP.Anlr.RefdBrB(*unit$*)**

Data Type Double Reference value.

Description This command sets the zero dB value of the Analog Analyzer dBr B reference.

Parameters

Name	Description
unit$	The following units are available: V, dBV, dBu

Example See example for `AP.Anlr.RefChAdBr`.

AP.Anlr.RefdBm ❶❷ Property

Syntax **AP.Anlr.RefdBm(*unit$*)**

Data Type Double Reference value.

Parameters

Name	Discription
unit$	The following units are available: Ohms.

Description This command sets the Analog Analyzer dBm impedance value. This value of circuit impedance is used as the "R" value in the equation V2/R to compute power from the measured voltage (V), followed by decibel conversion.

See Also `AP.Anlr.ChAImpedance, AP.Anlr.ChBImpedance.`

Example

```
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest  'Reset panels
   AP.Gen.ChBOutput = 0 'Set generator output B to OFF
   AP.Gen.Impedance = 2 'Set generator output Z to _
      600 Ohms
   AP.Gen.RefdBm("Ohms") = 600 'Set dBm reference to _
      600 Ohms
```

```
   If System = "1" Then
      AP.Gen.Ampl("dBm") = 0
   Else
      AP.Gen.ChAAmpl("dBm") = 0
   End If
   AP.Anlr.ChAInput = 0
   AP.Gen.Output = 1
   AP.Anlr.ChAImpedance = 1   'Set Cha A input Z to _
      600 ohms
   AP.Anlr.RefdBm("Ohms") = 600 'Set dBm reference To _
      600 Ohms
   Reference = AP.Anlr.RefdBm("Ohms")
   AP.Anlr.FuncSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.FuncTrig  'Trigger new reading.
   Do
      Ready = AP.Anlr.FuncReady 'Get status
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg("dBm") 'Get settled reading
   Debug.Print "Channel A Amplitude = ";Format(Reading, _
      "#.0000");" dBm (";Reference;" Ohms)"
End Sub
```

Example 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.

AP.Anlr.RefdBr

❶ **Property**

Syntax **AP.Anlr.RefdBr(*unit$*)**

Data Type Double — Reference value.

Parameters

Name	Description
unit$	The following units are available: V, dBV, dBu

Description

This command sets the zero dB value of the Analog Analyzer dBr reference.

Example

```
Sub Main
```

```
    AP.File.OpenTest ("SNR.AT1") 'Open signal-To-noise _
       test.
    AP.Anlr.RefdBr("V") = 5.0  'Set dBr reference to _
       5.0 Volts
    AP.Gen.Output = 0          'Turn generator OFF.
    AP.Sweep.Start             'Start Single point sweep.
End Sub
```

Comment This example performs a single point signal-to-noise measurement. The measurement displayed in the Data Editor is relative to 5.0 V.

AP.Anlr.RefdBrAuto

❶❷ Method

Syntax `AP.Anlr.RefdBrAuto`

Parameters None

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 the reference field for System One:

1 If the Function meter has dBr units selected the Function meter is used to determine the dBr Reference value.

2 If the Level meter has dBr units selected the Level meter is used to determine the dBr Reference value.

3 If the Function meter mode `AP.Anlr.FuncMode` is set to Amplitude (0), Bandpass (1), or Bandreject (2) mode the Function meter is used to determine the dBr Reference value.

4 If the Function meter mode `AP.Anlr.FuncMode` is set to THD+N (3 or 4) mode with the units set to Volts, dBm, dBu, dBV, or dBr the Function meter is used to determine the dBr Reference value.

5 Otherwise the Level meter is used to determine the dBr Reference value.

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 ❶

```
Sub Main
   AP.File.OpenTest ("SNR.AT1") 'Open signal-to-noise _
      test.
   AP.Anlr.RefdBrAuto          'Set dBr reference.
   Return = AP.Anlr.RefdBrAuto
   If Return = True Then Debug.Print "Reference Set"
   AP.Gen.Output = 0           'Turn generator OFF.
   AP.Sweep.Start              'Start single point sweep.
End Sub
```

Example Output

```
Reference Set
```

Comment

This example performs a single point signal to noise measurement. The measurement result is displayed in the Data Editor. The text "Reference Set" is output to the DeBug Immediate Tab of the Macro editor.

AP.Anlr.RefFreq

❶❷ Property

Syntax **AP.Anlr.RefFreq(*unit$*)**

Data Type Double — Set reference frequency value.

Parameters

Name	Description
unit$	The following units are available: Hz only.

Description This command sets the value of Analog Analyzer Frequency reference. This reference value is used by all Analog Analyzer relative frequency units (F/R, dHz, %Hz, cent, octs, decs, d%, dPPM).

Example

```
Sub Main
   AP.Application.NewTest        'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.RefFreq("Hz") = 1000
   Ref = AP.Anlr.RefFreq("Hz")
   AP.Anlr.ChAFreqSettling(.5, .0002, "Hz", 3, .03, 1)
   AP.Anlr.ChAFreqTrig           'Trigger new reading.
   Do
      Ready = AP.Anlr.ChAFreqReady  'Get status.
   Loop Until Ready > 0
   Reading = AP.Anlr.ChAFreqRdg("dHz") 'Get settled _
      reading.
   Debug.Print "Frequency delta relative to";Ref; _
      "Hz = ";Format(Reading, "#.0000");" dHz"
End Sub
```

Example Output

```
Frequency delta relative to 1000Hz = -0.3173 dHz
```

AP.Anlr.RefFreqAuto

02 Method

Syntax **AP.Anlr.RefFreqAuto**

Parameters None

Result Boolean

True	Frequency reference set.
False	Frequency reference NOT set.

Description This command sets the Analyzer Frequency Reference field to the current frequency reading.

Example

```
Sub Main
   AP.Application.NewTest     'Reset panels
   AP.Anlr.FuncInput = 0
   AP.Gen.Output = 1
```

```
   AP.Anlr.ChAInput = 1
   AP.Anlr.RefFreqAuto          'Set frequency reference.
   Return = AP.Anlr.RefFreqAuto  'Return reference _
      frequency.
   If Return = True Then Debug.Print "Reference Set"
   AP.Gen.Freq("Hz") = 2000
   AP.Anlr.ChAFreqTrig
   Do
      Ready = AP.Anlr.ChAFreqReady
   Loop Until Ready > 0
   A = AP.Anlr.ChAFreqRdg("dHz")
   Debug.Print "Frequency change = "; Format(A, "#.0000")
End Sub
```

Example Output

```
Reference Set
Frequency change = 999.6768
```

AP.Anlr.RefWatts

02 Property

Syntax **AP.Anlr.RefWatts(*unit$*)**

Data Type Double — Set Watts reference Impedance value.

Parameters

Name	Description
unit$	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 *VU!^X!O!A!2/R where R is the reference impedance.*

Example

```
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest        'Reset panels
   AP.Gen.Output = 1
   AP.Gen.RefWatts("Ohms") = 8
   If System = 1 Then
      AP.Gen.Ampl("W") = .1
      AP.Anlr.ChAInput = 1
   Else
      AP.Gen.ChAAmpl("W") = .1
```

```
      AP.Anlr.ChAInput = 2
   End If
   AP.Anlr.RefWatts(“Ohms”) = 8
   AP.Anlr.FuncSettling(1, .000002, “V”, 4, .05, 1)
   AP.Anlr.FuncTrig              ‘Trigger new reading.
   Do
      Ready = AP.Anlr.FuncReady  ‘Get status.
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg(“W”) ‘Get settled reading.
   Debug.Print “Output Power = ”;Format(Reading, _
      “#.0000");” Watts"
End Sub
```

Example Output

```
Output Power = .0997 Watts
```

AP.Anlr.WFDetector

❶❷ Property

Syntax `AP.Anlr.WFDetector`

Data Type Integer

0	NAB-RMS
1	JIS
2	IEC-PK

Description The Function meter mode must be set to Wow & Flutter.

See Also `AP.Anlr.FuncFilterWF, AP.Anlr.FuncMode`

Example

```
Sub Main
   AP.Application.NewTest        ‘Reset panels
   AP.Gen.Freq(“Hz”) = 3150
   AP.Gen.Ampl(“dBu”) = 0.0
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.FuncMode = 8
   AP.Anlr.FuncInput = 0
   AP.Anlr.WFDetector = 1     ‘Set W&F detector to JIS.
   AP.Anlr.WFFilter = 1 ‘Set W&F Filter to UnWeighted.
   AP.Anlr.FuncSettling(5, .0002, “%”, 3, .05, 1)
   AP.Anlr.FuncTrig           ‘Trigger new reading.
```

```
   Do
      Ready = AP.Anlr.FuncReady     'Get status.
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg("%")   'Get settled _
      reading.
   Debug.Print "Wow & Flutter = ";Format(Reading, _
      "#.0000");" %"
End Sub
```

Example Output Wow & Flutter = .0500 %

AP.Anlr.WFFilter ❶❷ Property

Syntax `AP.Anlr.WFFilter`

Data Type Integer

0	Weighted
1	UnWeighted
2	Weighted-High Band
3	UnWeighted-High Band
4	Wide-High Band
5	Scrape-High Band

Description This command sets the Analog Analyzer Wow & Flutter Filter weighting.

See Also `AP.Anlr.WFDetector, AP.Anlr.FuncMode`

Example See example for `AP.Anlr.WFDetector`.

User Notes

User Notes

User Notes

Application

AP.Application.AppDir ❶❷ Method

Syntax `AP.Application.AppDir`

Parameters None

Result String

Description

This command returns the application directory. When installing the software the default application directory is `"C:\APWINBIN\"` for Windows 3.1 and `"C:\Program Files\APWINBIN\"` for Windows 95. Utility programs are located in the application directory. Setting this command to a null string will display the Working Directory dialog.

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

Example Output

```
Long Path = C:\PROGRAM FILES\APWINBIN\
Short Path = C:\PROGRA~1\APWINBIN\
```

AP.Application.CopyPanelToClipboard ❶❷ Method

Syntax **AP.Application.CopyPanelToClipboard**

Data Type None

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

Example

```
Sub Main
   AP.Application.NewTest  'Create Graph with data
   AP.Gen.Output = True
   AP.Anlr.ChAInput = 2
   AP.Anlr.FuncFilterHP = 3
   AP.Anlr.FuncFilterLP = 0
   AP.Sweep.Data1.Id = 5906
   AP.Sweep.Source1.Start("Hz") = 50000.0
   AP.Sweep.Start
   AP.Graph.OptimizeLeft
   AP.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
```

AP.Application.DisplayDataOnTestOpen ❶❷ Property

Syntax **AP.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.

❶ Example

```
Sub Main
   AP.Application.DisplayDataOnTestOpen = 0
   AP.File.OpenTest "SAMPLE1.AT1"
'Define strings to be used in the following prompt.
   String1$ = "Test Loaded and data NOT displayed."
   AP.Prompt.Text = String1$
   AP.Prompt.FontSize = 10 'Set prompt font size to 8 _
      point.
   AP.Prompt.Position -1,-1,290,120       'Set prompt _
      location and size.
   AP.Prompt.ShowWithContinue    'Display prompt with _
      Continue Macro button displayed.
   Stop  'Stop Macro until Continue Macro _
      button is pressed.

   AP.Application.DisplayDataOnTestOpen = 1
   AP.File.OpenTest "SAMPLE1.AT1"
   'Define strings to be used in the following prompt.
   String1$ = "Test loaded and data displayed."
   AP.Prompt.Text = String1$
   AP.Prompt.FontSize = 10             'Set prompt _
      font size to 8 point.
   AP.Prompt.Position -1,-1,290,100    'Set prompt _
      location and size.
   AP.Prompt.ShowWithContinue    'Display prompt with _
      Continue Macro button displayed.
   Stop  'Stop Macro until Continue Macro _
      button is pressed.
End Sub
```

AP.Application.Input ❶❷ Method

Syntax **AP.Application.Input(***address%***)**

Parameters

Name	Description
address%	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 `AP.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 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 = AP.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
```

Example Output

```
All Lines pulled high.
```

AP.Application.MacroDir (OLE) ❶❷ Method

Syntax **AP.Application.MacroDir**

Parameters None

Result String

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

Example

```
Private Sub Form_Load()
    Dim AP As Object
    Set AP = CreateObject("APWIN.Application")
'The following lines makes the Visual Basic Current
'  Directory and the APWIN Working Directory the same
'  as the directory where the current APWIN macro was
'  loaded from.

    ChDir AP.Application.MacroDir
    AP.Application.WorkingDir = AP.Application.MacroDir

    'Your code goes here.

End Sub
```

AP.Application.MacroEditorVisible ❶❷ Method

Obsolete Obsolete command not recomended for new design.

Syntax `AP.Application.MacroEditorVisible`

Data Type Boolean

True	Restore Macro Editor to view.
False	Remove Macro Editor from view.

Result Void

Description This command when executed makes the APWIN Macro Editor visible or invisible.

Note: This command has been replaced by the `AP.Application.VisibleMacroEditor` command. The `AP.Application.MacroEditorVisible` command will continue to function as documented here but will be removed from all visible areas within APWIN.

AP.Application.Name ❶❷ Property

Syntax `AP.Application.Name`

Result String ASCII charactors.

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

Example

```
Sub Main
   AppActivate AP.Application.Name  'Select the APWIN _
      window
   'The following SendKey command will now be sent to _
      the APWIN application.
   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
```

AP.Application.NewData ❶❷ Method

Syntax **AP.Application.NewData**

Parameters None

Result Boolean

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

❶ **Example**

```
Sub Main
   AP.File.OpenTest "FRQ-RESP.AT1"     'Open frequency _
      response test.
   AP.Application.NewData
   AP.Sweep.Start                      'Start sweep.
   AP.File.SaveDataAs "FRQ-RESP.DAT"   'Save data.

   AP.File.OpenTest "THD-FRQ.AT1"      'Open total _
      harmonic distortion + noise test.
   AP.Application.NewData
   AP.Sweep.Start                      'Start sweep.
   AP.File.SaveDataAs "THD-FRQ.DAT"    'Save data.

   AP.File.OpenTest "RESIDNOI.AT1"     'Open residual _
      noise test.
```

```
   AP.Application.NewData
   AP.Sweep.Start                        'Start sweep.
   AP.File.SaveDataAs "RESIDNOI.DAT"     'Save data.
End Sub
```

AP.Application.NewMacro (OLE) ❶❷ Method

Syntax `AP.Application.NewMacro`

Parameters None

Result Boolean

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.

AP.Application.NewTest ❶❷ Method

Syntax `AP.Application.NewTest`

Parameters None

Result Boolean

Description This command initializes the current APWIN 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
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.FuncFilterLP = 1
   A = AP.Anlr.FuncFilterLP
   AP.Sweep.Data1.Id = 5906
   AP.Sweep.Source1.Start("Hz") = 100000
   AP.Sweep.Start
End Sub
```

AP.Application.Output

❶❷ Method

Syntax **AP.Application.Output(***address%, data%***)**

Parameters

Name	Description
address%	An I/O address value between 0 and 65535 (FFFF Hex).
data%	Any value between 0 and 255(FF Hex).

Result Void

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 NT operating system.

Example

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

AP.Application.Page

❶❷ Property

Syntax **AP.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
   AP.Application.NewTest  'Reset panels
   AP.Application.Page = 1
   Return = AP.Application.Page
   Debug.Print "Page "; Return; " displayed."
   Wait 1   'So the user can see the page change.
   AP.Application.Page = 2
   Return = AP.Application.Page
   Debug.Print "Page "; Return; " displayed."
   Wait 1
   AP.Application.Page = 3
   Return = AP.Application.Page
   Debug.Print "Page "; Return; " displayed."
End Sub
```


AP.Application.PanelClose

❶❷ Property

Syntax **AP.Application.PanelClose(***constant***)**

Data Type Long

Parameters

Constant	Description
apbPanelAnalogGenLarge	Remove the Analog Generator panel from view.
apbPanelAnalogGenSmall	Remove the Analog Generator panel from view.
apbPanelAnlrLarge	Remove the Analog Analyzer panel from view.
apbPanelAnlrSmall	Remove the Analog Analyzer panel from view.
apbPanelBarGraph?	Remove the desired Bar Graph 1 through 32 from view.
apbPanelDataEditor	Remove the Data Editor panel from view.

apbPanelDCXLarge

Remove the DCX-127 panel from view.

apbPanelDCXSmall

Remove the DCX-127 panel from view.

apbPanelDiagnostic

Remove the Diagnostic panel from view.

apbPanelDigIOLarge

Remove the Digital Input / Output panel from view.

apbPanelDigIOSmall

Remove the Digital Input / Output panel from view.

apbPanelDigitalAnlrLarge

Remove the Digital Analyzer panel from view.

apbPanelDigitalAnlrSmall

Remove the Digital Analyzer panel from view.

apbPanelDigitalGenLarge

Remove the Digital Generator panel from view.

apbPanelDigitalGenSmall

Remove the Digital Generator panel from view.

apbPanelDIOStatusBitsLarge

Remove the Status Bits panel from view.

apbPanelDIOStatusBitsSmall

Remove the Status Bits panel from view.

apbPanelDSPSmall

Remove the DSP panel from view.

apbPanelDSPLarge

Remove the DSP panel from view.

apbPanelGraph

Remove the Graph from view.

apbPanelRefInput

Remove the Sync Reference panel from view.

apbPanelRegulation

Remove the Regulation panel from view.

apbPanelSettling

Remove the Settling panel from view.

apbPanelSpeaker

Remove the Speaker panel from view.

apbPanelSweepSmall

Remove the Sweep panel from view.

apbPanelSweepLarge

Remove the Sweep panel from view.

apbPanelSwitcher

Remove the Switcher panel from view.

Description This command closes the selected panel,

Example

```
Sub Main
   AP.Application.NewTest
   AP.Application.PanelOpen(apbPanelAnalogGenLarge)
   AP.Application.PanelOpen(apbPanelAnlrLarge)
   AP.Application.PanelOpen(apbPanelSweep)
   AP.Gen.Output = True
   AP.Anlr.ChAInput = 2
   AP.Anlr.FuncFilterHP = 3
   AP.Sweep.Data1.Id = 5906
   AP.Application.Page = 2
   AP.Application.Page = 3
   AP.Application.PanelClose(apbPanelDigIOSmall)
   AP.Application.Page = 2
   AP.Sweep.Start
End Sub
```

AP.Application.PanelOpen

❶❷ Property

Syntax **AP.Application.PanelOpen(***constant***)**

Data Type Long

Parameters

Constant	Description
apbPanelAnalogGenLarge	Display Large view of Analog Generator panel.

apbPanelAnalogGenSmall

Display Large view of Analog Generator panel.

apbPanelAnlrLarge

Display Large view of Analog Analyzer panel.

apbPanelAnlrSmall

Display Small view of Analog Analyzer panel.

apbPanelBarGraph?

Display desired Bar Graph panel 1 through 32.

apbPanelDataEditor

Display Data Editor panel.

apbPanelDCXLarge

Display Large view of DCX-127 panel.

apbPanelDCXSmall

Display Small view of DCX-127 panel.

apbPanelDiagnostic

Display Diagnostic panel.

apbPanelDigIOLarge

Display Large view of Digital Input / Output panel.

apbPanelDigIOSmall

Display Small view of Digital Input / Output panel.

apbPanelDigitalAnlrLarge

Display Large view of Digital Analyzer panel.

apbPanelDigitalAnlrSmall

Display Small view of Digital Analyzer panel.

apbPanelDigitalGenLarge

Display Large view of Digital Generator panel.

apbPanelDigitalGenSmall

Display Small view of Digital Generator panel.

apbPanelDIOStatusBitsLarge

Display Large view of Status Bits panel.

apbPanelDIOStatusBitsSmall

Display Small view of Status Bits panel.

apbPanelDSPSmall

Display Small view of DSP panel.

apbPanelDSPLarge

Display Large view of DSP panel. When None is selected with the `AP.S1DSP.Program` or `AP.S2DSP.Program` commands this constant will display the Small view of the DSP panel.

apbPanelGraph

Display Graph panel.

apbPanelRefInput

Display Sync Reference panel.

apbPanelRegulation

Display Regulation panel.

apbPanelSettling

Display Settling panel.

apbPanelSpeaker

Display Speaker panel.

apbPanelSweepSmall

Display Small view of Sweep panel.

apbPanelSweepLarge

Display Large view of Sweep panel.

apbPanelSwitcher

Display Switcher panel.

Description This command displays the selected panel.

Example See example for `AP.Application.PanelClose`.

AP.Application.Quit ❶❷ Method

Syntax **`AP.Application.Quit`**

Parameters None

Result Void

Description This command terminates APWIN 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 APWIN 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 APWIN",.Field3
   End Dialog
   Dim Main_Menu As UserDialog

   Select Case Dialog(Main_Menu)
      Case 1
         'Incert your code here...
      Case Else
         AP.Application.Quit        'Exit APWIN
   End Select
   GoTo Start:
End Sub
```


AP.Application.Restore ❶❷ Method

Syntax `AP.Application.Restore`

Parameters None

Result Void

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
         AP.Application.Restore
      Case Else
         End
   End Select
   GoTo Start:
End Sub
```


AP.Application.SuppressErrorMessages ❶❷ Method

Syntax `AP.Application.MacroEditorVisible`

Data Type Boolean

True	Error messages not displayed.
False	(Default) Display Error messages.

Result Void

Description This command enables or disables display of Error Messages. Note: This command affects APWIN globaly. If set to True APWIN will not display any Error messages under manual operation or programatic control. When APWIN is loaded this command defaults to False so that all Error messages are displayed.

❷ **Example**

```
Sub Main
   With AP.App
      .SuppressErrorMessages = False   'Display Errors
      AP.File.OpenTest("xt5alk.at2")   'Errors displayed
      .SuppressErrorMessages = True 'Don't Display Errors
      AP.File.OpenTest("xt5alk.at2")'Errors NOT displayed
      .SuppressErrorMessages = False
End Sub
```

AP.Application.SysType ❶❷ Method

Syntax `AP.Application.SysType`

Parameters None

Result String

"1"	APWIN running in System One mode.
"2"	APWIN running in System Two mode.

Description This commandreturns the the current mode of the APWIN software.

Example

```
Sub Main
   System$ = AP.Application.SysType
   Select Case System$
      Case "1"
         AP.Prompt.Text = "APWIN configured for _
            System One hardware."
      Case "2"
         AP.Prompt.Text = "APWIN configured for _
            System Two hardware."
   End Select
   AP.Prompt.ShowWithContinue
   Stop
End Sub
```

AP.Application.TestDir ❶❷ Method

Syntax `AP.Application.TestDir`

Result String

Description This command returns the path of the test (.AT1 or .AT2) that is currently loaded.

❶ **Example**

```
Sub Main
   AP.Application.DisplayDataOnTestOpen = 0
   AP.File.OpenTest "SAMPLE1.AT1"
'Get current test name
```

```
   TestName$ = AP.Application.TestName
'Get directory that the current test was loaded from
   TestDir$ = AP.Application.TestDir

'Define strings to be used in the following prompt.
   String1$ = "Test file "
   String2$ = " was loaded from the "
   String3$ = " directory."
   AP.Prompt.Text = String1$ & TestName$ & String2$ & _
      TestDir$ & String3$
   AP.Prompt.FontSize = 10 'Set prompt font size to 8 _
      point.
   AP.Prompt.Position -1,-1,290,130    'Set prompt _
      location and size.
   AP.Prompt.ShowWithContinue    'Display prompt with _
      Continue Macro button displayed.
   Stop     'Stop Macro until Continue Macro _
      button is pressed.
End Sub
```


AP.Application.TestName

❶❷ Property

Syntax `AP.Application.TestName`

Result String

Description This command returns the test (.AT1 or .AT2) file name of the test that is currently loaded.

Example See example for `AP.Application.TestDir`.

AP.Application.Version

❶❷ Method

Syntax `AP.Application.VisibleMacroEditor`

Result Double

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

Example

```
Sub Main
   'APWIN version 1.5 required
   If AP.Application.Version <> 1.5 Then End
   AP.Application.NewTest
   AP.Application.PanelOpen(apbPanelAnalogGenLarge)
   AP.Application.PanelOpen(apbPanelAnlrLarge)
   AP.Application.PanelOpen(apbPanelSweep)
   AP.Gen.Output = True
   AP.Anlr.ChAInput = 2
   AP.Anlr.FuncFilterHP = 3
   AP.Sweep.Data1.Id = 5906
   AP.Application.Page = 2
   AP.Application.Page = 3
   AP.Application.PanelClose(apbPanelDigIOSmall)
   AP.Application.Page = 2
   AP.Sweep.Start
End Sub
```


AP.Application.Visible

❶❷ Property

Syntax `AP.Application.Visible`

Data Type Boolean

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

Result Void

Description This command when executed makes the APWIN window visable or invisible. The Macro Editor remains visible.

Example

```
Sub Main
   AP.Application.Visible = False 'Remove APWIN from view.
   AP.Application.NewTest         'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 2
```

```
    AP.Sweep.Start
    AP.Application.Visible = True 'Restore APWIN.
End Sub
```

AP.Application.VisibleAll ❶❷ Property

Syntax `AP.Application.VisibleAll`

Data Type Boolean

True	Restore to view.
False	Remove from view.

Result Void

Description This command enables or disables display of the Graph and Bar-Graph displays, Data Editor, and Panels when a test is loaded durring Macro execution only. Use this command at the beginning of your macro to decrease overall test times.

❷ **Example**

```
Private Sub Form_Load()
    Dim AP As Object
    Set AP = CreateObject("APWIN.Application")
' Create OLE link to APWIN.
    AP.Application.Visible = True ' Make APWIN visable

    AP.Application.VisibleAll = True
    AP.File.OpenTest "VIEW.AT2"
'Test loaded displaying ALL graphic panels
    AP.Application.VisibleBarGraphs = False
'Disable display of Bar Graphs
    AP.File.OpenTest "VIEW.AT2"
    AP.Application.VisibleDataEditor = False
    AP.File.OpenTest "VIEW.AT2"
'Disable display of Data Editor
    AP.Application.VisibleGraph = False
    AP.File.OpenTest "VIEW.AT2"
'Disable display of Graph
    AP.Application.VisiblePanels = False
    AP.File.OpenTest "VIEW.AT2"
'Disable display of Instrument panels
```

```
    AP.Application.Quit                    'Quit APWIN
    End
End Sub
```

AP.Application.VisibleBarGraphs ❶❷ Property

Syntax `AP.Application.VisibleBarGraphs`

Data Type Boolean

True	Restore to view.
False	Remove from view.

Result Void

Description This command enables or disables display of the Bar-Graph display, when a test is loaded durring Macro execution only. Use this command at the beginning of your macro to decrease overall test times.

Example See example for `AP.Application.VisibleAll`.

AP.Application.VisibleDataEditor ❶❷ Property

Syntax `AP.Application.VisibleDataEditor`

Data Type Boolean

True	Restore to view.
False	Remove from view.

Result Void

Description This command enables or disables display of the Data Editor panel when a test is loaded durring Macro execution only. Use this command at the beginning of your macro to decrease overall test times.

Example See example for `AP.Application.VisibleAll`.

AP.Application.VisibleGraph ❶❷ Property

Syntax `AP.Application.VisibleGraph`

Data Type Boolean

True	Restore to view.
False	Remove from view.

Result Void

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

Example See example for `AP.Application.VisibleAll`.

AP.Application.VisibleMacroEditor ❶❷ Method

Syntax `AP.Application.VisibleMacroEditor`

Data Type Boolean

True	Restore Macro Editor to view.
False	Remove Macro Editor from view.

Result Void

Description This command when executed makes the APWIN Macro Editor visible or invisible.

Example

```
Sub Main
   AP.Application.VisibleMacroEditor = False 'Remove _
      Macro Editor from view.
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 2
   AP.Sweep.Start
   AP.Application.VisibleMacroEditor = True 'Restore _
      Macro Editor.
End Sub
```

AP.Application.VisiblePanels

❶❷ Property

Syntax `AP.Application.VisiblePanels`

Data Type Boolean

True	Restore to view.
False	Remove from view.

Result Void

Description This command enables or disables display of the Panels when a test is loaded durring Macro execution only. Use this command at the beginninng of your macro to decrease overall test times.

Example See example for `AP.Application.VisibleAll`.

AP.Application.WorkingDir

❶❷ Method

Syntax `AP.Application.WorkingDir`

Parameters None

Result String

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

Example

```
Private Sub Form_Load()
   Dim AP As Object
   Set AP = CreateObject("APWIN.Application")
'The following line makes the APWIN Working Directory
'  the same as the VB current directory.
   If AP.Application.AppDir <> CurDir Then
      AP.Application.WorkingDir = CurDir

   'Your code goes here.

End Sub
```

User Notes

User Notes

User Notes

User Notes

User Notes

Bar Graph

AP.BarGraph.AxisAutoScale ❶❷ Property

Syntax **AP.BarGraph.AxisAutoScale(*barid%*)**

Data Type Boolean

Parameters

Name	Description
barid%	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
   AP.Application.NewTest        'Setup Code to make _
      something to test
   AP.Gen.Output = True
   AP.Gen.ChAFreq("Hz") = 3000.0
   AP.Anlr.ChAInput = 2
   AP.Anlr.FuncMode = 1
   AP.Anlr.FuncBPBRTuning = 4
   AP.Anlr.FuncBPBRFreq("Hz") = 3000.0 'End Setup code

   With AP.BarGraph
      GenFreqBar = .New          'Create New Bargraph
      .Id(GenFreqBar)  = 5051    'Configure Bargraph _
         to control Generator Frequency
      .AxisLogLin(GenFreqBar) = 1         'Linear axis
      .AxisRight(GenFreqBar,"Hz") = 3500.0 'Right value
      .AxisLeft(GenFreqBar,"Hz") = 2500.0 'Left value
      .AxisIncrement(GenFreqBar,"Hz") = 10.0 'Step size

      AnlrFuncRdg = .New         'Create New Bargraph
      .Id(AnlrFuncRdg) = 5907    'Configure Bargraph _
         to display Function meter readings
      .DigitsOnly(AnlrFuncRdg) = False 'Display Digits _
         and Bar on the Bargraph
      .AxisLogLin(AnlrFuncRdg) = 1     'Linear axis
      .AxisLeft(AnlrFuncRdg,"V") = 0.8    'Left value
```

```
    .AxisRight(AnlrFuncRdg,"V") = 1.2   'Right value
    .AxisAutoScale(AnlrFuncRdg) = True  'Autoscale _
       Readings
    .TargetLower(AnlrFuncRdg,"V") = 0.95   'Target _
       Lower value
    .TargetUpper(AnlrFuncRdg,"V") = 1.05   'Target _
       Upper value
    .TargetRange(AnlrFuncRdg) = True       'Display _
       Target area

    .Reset(GenFreqBar)   'Reset #1 Min/Max readings
    .Reset(AnlrFuncRdg)  'Reset #2 Min/Max readings

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

    GenMaxSet = .Max(GenFreqBar)  'Create _
       readings prompt
    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$(AP.Gen.Freq("Hz")),6) & " Hz" & Chr(13)
```

```
   CurRdg$ = "Current Amplitude Reading = " _
      & Left(Str$(AP.Anlr.FuncRdg("V")),6) & " V"

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

AP.BarGraph.AxisIncrement

❶❷ Property

Syntax **AP.BarGraph.AxisIncrement(***barid%, "unit$"***)**

Data Type Double

Parameters

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 AP.BarGraph.Id 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 AP.BarGraph.AxisAutoScale.

AP.BarGraph.AxisLeft

❶❷ Property

Syntax **AP.BarGraph.AxisLeft(***barid%, "unit$"***)**

Data Type Double

Parameters

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 `AP.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 `AP.BarGraph.AxisRight, AP.BarGraph.AxisAutoScale`

Example See example for `AP.BarGraph.AxisAutoScale`.

AP.BarGraph.AxisLogLin

❶❷ Property

Syntax **`AP.BarGraph.AxisLogLin(`*barid%*`)`**

Data Type Integer

0	Logarithmic axis.
1	Linear axis.

Parameters

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 `AP.BarGraph.AxisAutoScale`.

AP.BarGraph.AxisRight

❶❷ Property

Syntax **`AP.BarGraph.AxisRight(`*barid%*`)`**

Data Type Double

Parameters

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 `AP.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 `AP.BarGraph.AxisLeft, AP.BarGraph.AxisAutoScale`

Example See example for `AP.BarGraph.AxisAutoScale`.

AP.BarGraph.DigitsOnly ❶❷ Property

Syntax **`AP.BarGraph.DigitsOnly(`*`barid%`*`)`**

Data Type Boolean

`True`	Display digits only.
`False`	Display digits and Bar Graph.

Parameters

Name	Description
`barid%`	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 charactors) or the digits and the bar on the Bargraph.

Example See example for `AP.BarGraph.AxisAutoScale`.

AP.BarGraph.Id ❶❷ Property

Syntax **`AP.BarGraph.Id(`*`barid%`*`)`**

Data Type Long Instrument Parameter ID#.

Parameters

Name	Description
`barid%`	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 D to obtain instrument parameter identification numbers.

Example See example for `AP.BarGraph.AxisAutoScale`.

AP.BarGraph.Max

❶❷ Property

Syntax `AP.BarGraph.Max(`*barid%, "unit$"*`)`

Parameters

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 `AP.BarGraph.Id` command to determine the appropriate unit selections.

Result Double

Description This command returns the maximum measured value obtained durring the time sence the last reset for the selected Bar Graph

See Also `AP.BarGraph.Reset, AP.BarGraph.Min`

Example See example for `AP.BarGraph.AxisAutoScale`.

AP.BarGraph.Min

❶❷ Property

Syntax `AP.BarGraph.Min(`*barid%, "unit$"*`)`

Parameters

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 `AP.BarGraph.Id` command to determine the appropriate unit selections.

Result Double

Description This command returns the maximum measured value obtained durring the time sence the last reset for the selected Bar Graph

See Also `AP.BarGraph.Reset, AP.BarGraph.Max`

Example See example for `AP.BarGraph.AxisAutoScale`.

AP.BarGraph.New

01 02 Method

Syntax `AP.BarGraph.New[(id%)]`

Parameters

Name	Description
id%	Instrument identification number. Refer to Appendix D to obtain instrument parameter identification numbers.

Result Integer

1-32	Identification number of Bar Graph created.

Description This command creates a new Bar Graph and returns the identification number.

Example See example for `AP.BarGraph.AxisAutoScale`.

AP.BarGraph.Reset

01 02 Method

Syntax `AP.BarGraph.Reset(barid%)`

Parameters

Name	Description
barid%	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 `AP.BarGraph.Max, AP.BarGraph.Min`

Example See example for `AP.BarGraph.AxisAutoScale`.

AP.BarGraph.TargetLower

01 02 Property

Syntax `AP.BarGraph.TargetLower(barid%, "unit$")`

Data Type Double

Parameters

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 `AP.BarGraph.Id` command to determine the appropriate unit selections.

Description This command defines the target value for the left side of the Bar Graph.

See Also `AP.BarGraph.TargetUpper, AP.BarGraph.TargetRange`

Example See example for `AP.BarGraph.AxisAutoScale`.

AP.BarGraph.TargetRange

❶❷ Property

Syntax **`AP.BarGraph.TargetRange(`***barid%***`)`**

Data Type Boolean

True	Target area displayed.
False	Target area not displayed.

Parameters

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

Description This command turns the selected Bar Graph Target Range ON or OFF.

See Also `AP.BarGraph.TargetLowe, AP.BarGraph.TargetUpper`

Example See example for `AP.BarGraph.AxisAutoScale`.

AP.BarGraph.TargetUpper

❶❷ Property

Syntax **`AP.BarGraph.TargetUpper(`***barid%, "unit$"***`)`**

Data Type Double

Parameters

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 `AP.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 `AP.BarGraph.TargetLower, AP.BarGraph.TargetRange`

Example See example for `AP.BarGraph.AxisAutoScale`.

User Notes

User Notes

User Notes

User Notes

User Notes

Status Bits

AP.Bits.ChAAudioModeRdg ❶❷ Property

Syntax **AP.Bits.ChAAudioModeRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Audio Mode
1	Data Mode

Description This command returns the Status Bits channel A 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 AP.Bits.ChAStatusXferToString command.

See Also AP.Bits.ChAStatusXferToString

Example

```
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest  'Reset panels
   If System = "1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .3 'Wait for reading to update
   Debug.Print "Audio Mode = " & AP.Bits.ChAAudioModeRdg
End Sub
```

Example Output

```
Audio Mode = 0
```

AP.Bits.ChAAuxBitsRdg

12 Property

Syntax `AP.Bits.ChAAuxBitsRdg([string$])`

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	20-bit not defined
1	24-bit not defined
2	20-bit single
3	Reserved

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   Dim String_Array(3)
   String_Array(0) = “20-bit not defined”
   String_Array(1) = “24-bit main audio”
   String_Array(2) = “20-bit single”
   String_Array(3) = “Reserved”

   System = AP.Application.SysType
   AP.Application.NewTest  ‘Reset panels
   AP.Bits.Mode = 1        ‘Professional Mode
   If System = “1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .3 ‘Wait for reading to update
```

```
   Debug.Print "Auxiliary Bits Reading = " & _
      String_Array(AP.Bits.ChAAuxBitsRdg)
End Sub
```

Example Output

```
Auxiliary Bits Reading = 20-bit not defined
```


AP.Bits.ChACategoryRdg ❶❷ Property

Syntax `AP.Bits.ChACategoryRdg([string$])`

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	General
1	CD Player
2	PCM Adaptor
3	DAT Recorder
4	Digital Broadcast
5	Musical Instrument

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   Dim String_Array(5)
   String_Array(0) = "General"
   String_Array(1) = "CD Player"
   String_Array(2) = "PCM Adaptor"
   String_Array(3) = "DAT Recorder"
```

```
   String_Array(4) = "Digital Broadcast"
   String_Array(5) = "Musical Instrument"

   System = AP.Application.SysType
   AP.Application.NewTest  'Reset panels
   If System = "1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .3  'Wait for reading to update
   Debug.Print "Category Reading = " & _
      'String_Array(AP.Bits.ChACategoryRdg)
End Sub
```

Example Output

```
Category Reading = General
```

AP.Bits.ChAChModeRdg

02 Property

Syntax **AP.Bits.ChAChModeRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Not Indicated
1	2-channel
2	Single-channal
3	Primary/Sec
4	Stereo
5	Reserved-1
6	Reserved-2
7	Vector to byte 3

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   Dim String_Array(7)
   String_Array(0) = “Not Indicated”
   String_Array(1) = “2-channel”
   String_Array(2) = “Single-channal”
   String_Array(3) = “Primary/Sec”
   String_Array(4) = “Stereo”
   String_Array(5) = “Reserved-1"
   String_Array(6) = “Reserved-2"
   String_Array(7) = “Vector to byte 3"

   System = AP.Application.SysType
   AP.Application.NewTest  ‘Reset panels
   AP.Bits.Mode = 1     ‘Professional Mode
   If System = “1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .3  ‘Wait for reading to update
   Debug.Print “Channel Mode Reading = ” & _
      String_Array(AP.Bits.ChAChModeRdg)
End Sub
```

Example Output

```
Channel Mode Reading = Not Indicated
```

AP.Bits.ChAChNumRdg ❶❷ Property

Syntax **`AP.Bits.ChAChNumRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
`string$`	Optional string containing status bit information.

Result

Value	Description
0	Don't Care
1	A (Left)
2	B (Right)
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O

Description

This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also

`AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   Dim String_Array(15)
   String_Array(0)= "Don't Care"
   String_Array(1) = "A (Left)"
   String_Array(2) = "B (Right)"
   String_Array(3) = "C"
   String_Array(4) = "D"
   String_Array(5) = "E"
   String_Array(6) = "F":
   String_Array(7) = "G"
   String_Array(8) = "H"
   String_Array(9) = "I"
```

```
   String_Array(10) = "J"
   String_Array(11) = "K"
   String_Array(12) = "L"
   String_Array(13) = "M"
   String_Array(14) = "N"
   String_Array(15) = "O"

   System = AP.Application.SysType
   AP.Application.NewTest  'Reset panels
   AP.Bits.Mode = 0     'Consumer Mode
   If System = "1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .5  'Wait for reading to update
   Debug.Print "Channel Number Reading = " & _
      String_Array(AP.Bits.ChAChNumRdg)
End Sub
```

Example Output

```
Channel Number Reading = Don't Care
```

AP.Bits.ChAClockAccuracyRdg ❶❷ Property

Syntax **AP.Bits.ChAClockAccuracyRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Level 1
1	Level 2
2	Level 3

3	Reserved

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   Dim String_Array(3)
   String_Array(0)= “Level 1"
   String_Array(1) = “Level 2"
   String_Array(2) = “Level 3"
   String_Array(3) = “Reserved”

   System = AP.Application.SysType
   AP.Application.NewTest  ‘Reset panels
   AP.Bits.Mode = 0     ‘Consumer Mode
   If System = “1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .5  ‘Wait for reading to update
   Debug.Print “Clock Accuracy Reading = ” & _
      String_Array(AP.Bits.ChAClockAccuracyRdg)
End Sub
```

Example Output

```
Clock Accuracy Reading = Level 2
```

AP.Bits.ChACopyrightRdg

❶❷ Property

Syntax **`AP.Bits.ChACopyrightRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
`string$`	Optional string containing status bit information.

Result

Value	Description
0	Copyright
1	Non-Copyright

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   Dim String_Array(1)
   String_Array(0)= “Copyright”
   String_Array(1) = “Non-Copyright”

   System = AP.Application.SysType
   AP.Application.NewTest  ‘Reset panels
   AP.Bits.Mode = 0     ‘Consumer Mode
   If System = “1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .5  ‘Wait for reading to update
   Debug.Print “Copyright Reading = ” & _
      String_Array(AP.Bits.ChACopyrightRdg) & _
      “ protected.”
End Sub
```

Example Output

```
Copyright Reading = Copyright protected.
```

AP.Bits.ChACrcRdg ❷ Property

Syntax **`AP.Bits.ChACrcRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Invalid
1	Valid

Description

This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also

`AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   System = AP.App.SysType
   If System = “1" Then
      Debug.Print"This example is for System Two only."
      End
   End If
   Dim String_Array(1)
   String_Array(0)= “Clear”
   String_Array(1) = “Set”
   System = AP.App.SysType
   AP.App.NewTest
   AP.Bits.Mode = 1                    ‘Professional Mode
   If System = “1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .5  ‘Wait for reading to update
      Debug.Print “Crc Valid Reading = ” & _
         String_Array(AP.Bits.ChACrcRdg)
End Sub
```

AP.Bits.ChADestinationRdg ❶❷ Property

Syntax `AP.Bits.ChADestinationRdg([string$])`

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

```
Example
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest  'Reset panels
   AP.Bits.Mode = 1     'Professional Mode
   If System = "1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   AP.Bits.Pro.Destination = "ABCD"
   Wait .5  'Wait for reading to update
   Debug.Print "Destination Reading = " & _
      AP.Bits.ChADestinationRdg
End Sub
```

Example Output

```
Destination Reading = ABCD
```

AP.Bits.ChAEmphRdg ❶❷ Property

Syntax `AP.Bits.ChAEmphRdg([string$])`

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
The following list is for Consumer Mode.	
0	No Pre-emph
1	50/15S
The following list is for Professional Mode.	
0	Not Indicated
1	None
2	50/15 uS
3	CCITT J.17

Description

This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also

`AP.Bits.ChAStatusXferToString, AP.Bits.ChAModeRdg`

Example

```
Sub Main
   Dim String_Array_Cons(1)
   Dim String_Array_Pro(3)
   String_Array_Cons(0) = “No Pre-emph”
   String_Array_Cons(1) = “50/15S”

   String_Array_Pro(0) = “Not Indicated”
   String_Array_Pro(1) = “None”
   String_Array_Pro(2) = “50/15S”
   String_Array_Pro(3) = “CCITT J.17"

   System = AP.Application.SysType
   AP.Application.NewTest  ‘Reset panels

   With AP.Bits
      .XmitChannel = 0
```

```
            .Mode = 0       'Consumer Mode
            .Cons.Emphasis = 1
            .XmitChannel = 1
            .Mode = 1       'Professional Mode
            .Pro.Emphasis = 3
         End With

         If System = "1" Then
            AP.S1Dio.InFormat = 2
         Else
            AP.S2Dio.InFormat = 3
         End If
         Wait 1   'Wait for reading to update
         With AP.Bits
            If .ChAModeRdg = 0 Then
               Debug.Print "Ch A Consumer Emphasis Reading = " _
               & String_Array_Cons(.ChAEmphRdg)
            Else
               Debug.Print "Ch A Professional Emphasis _
               Reading = " & String_Array_Pro(.ChAEmphRdg)
            End If

            If System = "2" Then
               If .ChBModeRdg = 0 Then
                  Debug.Print "Ch B Consumer Emphasis _
                  Reading = " & String_Array_Cons(.ChBEmphRdg)
               Else
                  Debug.Print "Ch B Professional Emphasis _
                  Reading = " & String_Array_Pro(.ChBEmphRdg)
               End If
            End If
         End With
      End Sub
```

Example Output

```
Ch A Consumer Emphasis Reading = 50/15S
Ch B Professional Emphasis Reading = CCITT J.17
```

AP.Bits.ChAFlag0_5Rdg

12 Property

Syntax **AP.Bits.ChAFlag0_5Rdg([*string$*])**

Data Type Boolean

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
True	Set
False	Clear

Description This command returns the Status Bits channel A Flag 0-5 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 AP.Bits.ChAStatusXferToString command.

See Also AP.Bits.ChAStatusXferToString

Example

```
Sub Main
   Dim String_Array(1)
   String_Array(0)= “Cleared”
   String_Array(1) = “Set”

   System = AP.Application.SysType
   AP.Application.NewTest  ‘Reset panels
   AP.Bits.Mode = 1     ‘Professional Mode
   If System = “1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   AP.Bits.Pro.Flag14_17 = 1  ‘Set flags 14-17
   AP.Bits.Pro.Flag18_21 = 1  ‘Set Flags 18-21
   Wait .5  ‘Wait for reading to update
   Debug.Print “Reliability Flags 0-5 Reading = ” & _
      String_Array(AP.Bits.ChAFlag0_5Rdg)
   Debug.Print “Reliability Flags 6-13 Reading = ” & _
```

```
        String_Array(AP.Bits.ChAFlag6_13Rdg)
    Debug.Print "Reliability Flags 14-17 Reading = " & _
        String_Array(AP.Bits.ChAFlag14_17Rdg)
    Debug.Print "Reliability Flags 18-21 Reading = " & _
        String_Array(AP.Bits.ChAFlag18_21Rdg)
End Sub
```

Example 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
```

AP.Bits.ChAFlag6_13Rdg

❶❷ Property

Syntax **AP.Bits.ChAFlag6_13Rdg([*string$*])**

Data Type Boolean

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
True	Set
False	Clear

Description This command returns the Status Bits channel A Flag 6-13 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example See example for `AP.Bits.ChAFlag0_5Rdg`.

AP.Bits.ChAFlag14_17Rdg

❶❷ Property

Syntax **AP.Bits.ChAFlag14_17Rdg([*string$*])**

Data Type Boolean

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
True	Set
False	Clear

Description This command returns the Status Bits channel A Flag 14-17 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example See example for `AP.Bits.ChAFlag0_5Rdg`.

AP.Bits.ChAFlag18_21Rdg

❶❷ Property

Syntax **`AP.Bits.ChAFlag18_21Rdg([`*`string$`*`])`**

Data Type Boolean

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
True	Set
False	Cleared

Description This command returns the Status Bits channel A Flag 18-21 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example See example for `AP.Bits.ChAFlag0_5Rdg`.

AP.Bits.ChAFreqModeRdg ❶❷ Property

Syntax **`AP.Bits.ChAFreqModeRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
`string$`	Optional string containing status bit information.

Result

Value	Description
`0`	Unlocked
`1`	Locked

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   Dim String_Array(3)
   String_Array(0) = “Unlocked”
   String_Array(1) = “Locked”

   System = AP.Application.SysType
   AP.Application.NewTest  ‘Reset panels
   AP.Bits.Mode = 1     ‘Professional Mode
   If System = “1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
```

```
   End If
   Wait 1 'Wait for reading to update
   Debug.Print "Frequency Mode Reading = " & _
      String_Array(AP.Bits.ChAFreqModeRdg)
End Sub
```

Example Output

```
Frequency Mode Reading = Locked
```

AP.Bits.ChALocalAddressRdg

❶❷ Property

Syntax **AP.Bits.ChALocalAddressRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest
   AP.Bits.Mode = 1     'Professional Mode
   If System = "1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   AP.Bits.XmitChannel = 0
   AP.Bits.Pro.LocalAddress = False
   AP.Bits.Pro.LocalAddress = 1234     'Set Ch A
   If System = "2" Then
      AP.Bits.XmitChannel = 1
      AP.Bits.Pro.LocalAddress = 5678  'Set Ch B
```

```
   End If
   Wait .5  'Wait for reading to update
   Debug.Print "Ch A Origin Reading = " & _
      AP.Bits.ChALocalAddressRdg
   If System = "2" Then
      Debug.Print "Ch B Local Address Reading = " & _
         AP.Bits.ChBLocalAddressRdg
   End If
End Sub
```

AP.Bits.ChAModeRdg ❶❷ Property

Syntax **AP.Bits.ChAModeRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Consumer
1	Professional

Description This command returns the Status Bits channel A 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 AP.Bits.ChAStatusXferToString command.

See Also AP.Bits.ChAStatusXferToString

Example

```
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest  'Reset panels
   If System = "1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
```

```
   AP.Bits.Mode = 0
   For LoopNum = 1 To 5 Step 1
      Debug.Print "Mode = " & AP.Bits.ChAModeRdg()
      If LoopNum = 3 Then
         AP.Bits.Mode = 1
         Wait .5
      End If
   Next LoopNum
End Sub
```

Example Output

```
Mode = 0
Mode = 0
Mode = 0
Mode = 1
Mode = 1
```

AP.Bits.ChAOriginRdg

❶❷ Property

Syntax **AP.Bits.ChAOriginRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

```
Example
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest  'Reset panels
   AP.Bits.Mode = 1     'Professional Mode
   If System = "1" Then
      AP.S1Dio.InFormat = 2
   Else
```

```
      AP.S2Dio.InFormat = 3
   End If
   AP.Bits.XmitChannel = 0
   AP.Bits.Pro.Origin = "ABCD"      'Set Ch A
   If System = "2" Then
      AP.Bits.XmitChannel = 1
      AP.Bits.Pro.Origin = "1234"   'Set Ch B
   End If
   Wait .5  'Wait for reading to update
   Debug.Print "Ch A Origin Reading = " & _
      AP.Bits.ChAOriginRdg
   If System = "2" Then
      Debug.Print "Ch B Origin Reading = " & _
         AP.Bits.ChBOriginRdg
   End If
End Sub
```

Example Output

```
Ch A Origin Reading = ABCD
Ch B Origin Reading = 1234
```

AP.Bits.ChARefSignalRdg

❶❷ Property

Syntax **AP.Bits.ChARefSignalRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Not a ref. Signal
1	Grade 1
2	Grade 2
3	Reserved

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   Dim String_Array(3)
   String_Array(0) = “Not a ref. Signal”
   String_Array(1) = “Grade 1"
   String_Array(2) = “Grade 2"
   String_Array(3) = “Reserved”

   System = AP.Application.SysType
   AP.Application.NewTest  ‘Reset panels
   AP.Bits.Mode = 1     ‘Professional Mode
   If System = “1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .5  ‘Wait for reading to update
   Debug.Print “Reference Signal Reading = ” & _
      String_Array(AP.Bits.ChARefSignalRdg)
End Sub
```

Example Output

```
Reference Signal Reading = Not a ref. Signal
```

AP.Bits.ChASampleFreqRdg ❶❷ Property

Syntax **`AP.Bits.ChASampleFreqRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
`string$`	Optional string containing status bit information.

Result

Value	Description

The following list is for Consumer Mode.

0	48 kHz
1	44.1 kHz
2	32 kHz

The following list is for Professional Mode.

0	Not Indicated
1	48 kHz
2	44.1 kHz
3	32 kHz

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   Dim String_Array_Cons(2)
   Dim String_Array_Pro(3)
   String_Array_Cons(0) = “48 kHz”
   String_Array_Cons(1) = “44.1 kHz”
   String_Array_Cons(2) = “32 kHz”

   String_Array_Pro(0) = “Not Indicated”
   String_Array_Pro(1) = “48 kHz”
   String_Array_Pro(2) = “44.1 kHz”
   String_Array_Pro(3) = “32 kHz”

   System = AP.Application.SysType
   AP.Application.NewTest  ‘Reset panels

   With AP.Bits
      .XmitChannel = 0
      .Mode = 0            ‘Consumer Mode
      .Cons.SampleFreq = 0
      .XmitChannel = 1
      .Mode = 1            ‘Professional Mode
      .Pro.SampleFreq = 2
```

```
            End With

            If System = "1" Then
               AP.S1Dio.InFormat = 2
            Else
               AP.S2Dio.InFormat = 3
            End If
            Wait 1   'Wait for reading to update
            With AP.Bits
               If .ChAModeRdg = 0 Then
                  Debug.Print "Ch A Consumer Frequency Reading _
                     = " & String_Array_Cons(.ChASampleFreqRdg)
               Else
                  Debug.Print "Ch A Professional Frequency _
                     Reading = " & String_Array_Pro _
                     (.ChASampleFreqRdg)
               End If

               If System = "2" Then
                  If .ChBModeRdg = 0 Then
                     Debug.Print "Ch B Consumer Frequency _
                        Reading = " & String_Array_Cons _
                        (.ChBSampleFreqRdg)
                  Else
                     Debug.Print "Ch B Professional Frequency _
                        Reading = " & String_Array_Pro _
                        (.ChBSampleFreqRdg)
                  End If
               End If
            End With
         End Sub
```

Example Output

```
Ch A Consumer Frequency Reading = 48 kHz
Ch B Professional Frequency Reading = 44.1 kHz
```

AP.Bits.ChASourceNumRdg

❶❷ Property

Syntax **AP.Bits.ChASourceNumRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Don't Care
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16

Description

This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also

`AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   Dim String_Array(0)
   String_Array(0)= "Don't Care"

   System = AP.Application.SysType
   AP.Application.NewTest  'Reset panels
```

```
   AP.Bits.Mode = 0     'Consumer Mode
   If System = "1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   AP.Bits.Cons.SourceNum = 5
   Wait .5  'Wait for reading to update
   If AP.Bits.ChASourceNumRdg = 0 Then
      Debug.Print "Source Number Reading = " & _
         String_Array(AP.Bits.ChASourceNumRdg)
   Else
      Debug.Print "Source Number Reading = " & _
         AP.Bits.ChASourceNumRdg
   End If
End Sub
```

Example Output

```
Source Number Reading = 5
```

AP.Bits.ChAStatusXferToArray

02 Property

Obsolete Obsolete command not recomended for new design.

Syntax `AP.Bits.ChAStatusXferToArray`

Data Type Variant

Parameters None

Description This command transfers the contents of the channel A Status Bits into an string. This enable the programmer to extract all of the status information from an single measurement.

Note: This command has been replaced by the `AP.Bits.ChAStatusXferToString` command. The `AP.Bits.ChAStatusXferToArray` command will continue to function as documented here but will be removed from all visible areas within APWIN.

AP.Bits.ChAStatusXferToString ❶❷ Property

Syntax **AP.Bits.ChAStatusXferToString**

Data Type Variant

Parameters None

Description This command transfers the contents of the channel A Status Bits into an string. This enable the programmer to extract all of the status information from an single measurement.

See Also AP.Bits.ChAXmitData

Example

```
Sub Main
   With AP.Bits
      Channel_A_Status = .ChAStatusXferToString
      Mode = .ChAModeRdg(Channel_A_Status)
      Debug.Print "Mode = " & Mode

      If Mode = 0 Then  'Consumer
         Debug.Print "Audio Mode = " & _
            .ChAAudioModeRdg(Channel_A_Status)
         Debug.Print "Copyright = " & _
            .ChACopyrightRdg(Channel_A_Status)
         Debug.Print "Emphasis = " & _
            .ChAEmphRdg(Channel_A_Status)
         Debug.Print "Channel Mode = " & _
            .ChAChModeRdg(Channel_A_Status)
         Debug.Print "Category Code = " & _
            .ChACategoryRdg(Channel_A_Status)
         Debug.Print "Source Number = " & _
            .ChASourceNumRdg(Channel_A_Status)
         Debug.Print "Channel Number = " & _
            .ChAChNumRdg(Channel_A_Status)
         Debug.Print "Sample Frequency = " & _
            .ChASampleFreqRdg(Channel_A_Status)
         Debug.Print "Clock Accuracy = " & _
            .ChAClockAccuracyRdg(Channel_A_Status)
      Else           'Professional
         Debug.Print "Audio Mode = " & _
            .ChAAudioModeRdg(Channel_A_Status)
```

```
            Debug.Print "Emphasis = " & _
               .ChAEmphRdg(Channel_A_Status)
            Debug.Print "Frequency Mode = " & _
               .ChAFreqModeRdg(Channel_A_Status)
            Debug.Print "Sample Frequency = " & _
               .ChASampleFreqRdg(Channel_A_Status)
            Debug.Print "Channel Mode = " & _
               .ChAChModeRdg(Channel_A_Status)
            Debug.Print "User Bits = " & _
               .ChAUserBitsRdg(Channel_A_Status)
            Debug.Print "Aux Bits = " & _
               .ChAAuxBitsRdg(Channel_A_Status)
            Debug.Print "Word Length = " & _
               .ChAWordLengthRdg(Channel_A_Status)
            Debug.Print "Ref Signal = " & _
               .ChARefSignalRdg(Channel_A_Status)
            Debug.Print "Origin Code = " & _
               .ChAOriginRdg(Channel_A_Status)
            Debug.Print "Destination Code = " & _
               .ChADestinationRdg(Channel_A_Status)
            Debug.Print "Local Address = " & _
               .ChALocalAddressRdg(Channel_A_Status)
            Debug.Print "Time Of Day = " & _
               .ChATimeOfDayRdg(Channel_A_Status)
            Debug.Print "Flag 0-5 = " & _
               .ChAFlag0_5Rdg(Channel_A_Status)
            Debug.Print "Flag 6-13 = " & _
               .ChAFlag6_13Rdg(Channel_A_Status)
            Debug.Print "Flag 14-17 = " & _
               .ChAFlag14_17Rdg(Channel_A_Status)
            Debug.Print "Flag 18-21 = " & _
               .ChAFlag18_21Rdg(Channel_A_Status)
            Debug.Print "Crc Valid = " & _
               .ChACrcRdg(Channel_A_Status)
         End If
      End With
   End Sub
```

AP.Bits.ChATimeOfDayRdg ❶❷ Property

Syntax **AP.Bits.ChATimeOfDayRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Description This command returns the Status Bits channel A 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 AP.Bits.ChAStatusXferToString command.

See Also AP.Bits.ChAStatusXferToString

Example

```
Sub Main
   System = AP.Application.SysType
   AP.Application.NewTest  'Reset panels
   AP.Bits.Mode = 1     'Professional Mode
   AP.Bits.Pro.LocalAddressAuto = 0
   AP.Bits.Pro.TimeOfDay = 123456789
   If System = "1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .5  'Wait for reading to update
   Debug.Print "Ch A Time Of Day Reading = " & _
      AP.Bits.ChATimeOfDayRdg
   If System = "2" Then
      Debug.Print "Ch B Time Of Day Reading = " & _
         AP.Bits.ChBTimeOfDayRdg
   End If
End Sub
```

Example Output

```
Ch A Time Of Day Reading = 123456789
Ch B Time Of Day Reading = 123456789
```

AP.Bits.ChAUserBitsRdg

12 Property

Syntax `AP.Bits.ChAUserBitsRdg([string$])`

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	None
1	192-bit block
2	Reserved
3	User defined

Description This command returns the Status Bits channel A 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 `AP.Bits.ChAStatusXferToString` command.

See Also `AP.Bits.ChAStatusXferToString`

Example

```
Sub Main
   Dim String_Array(3)
   String_Array(0)= “None”
   String_Array(1)= “192-bit block”
   String_Array(2)= “Reserved”
   String_Array(3)= “User defined”

   System = AP.Application.SysType
   AP.Application.NewTest  ‘Reset panels
   AP.Bits.Mode = 1     ‘Professional Mode
   AP.Bits.Pro.UserBits = 1
   If System = “1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .5  ‘Wait for reading to update
```

```
   Debug.Print "User Bits Reading = " & _
      String_Array(AP.Bits.ChAUserBitsRdg)
End Sub
```

Example Output

```
User Bits Reading = 192-bit block
```

AP.Bits.ChAWordLengthRdg ❶❷ Property

Syntax **AP.Bits.ChAWordLengthRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Not Indicated
1	20 bits
2	19 bits
3	18 bits
4	17 bits
5	16 bits

Description This command returns the Status Bits channel A 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 AP.Bits.ChAStatusXferToString command.

See Also AP.Bits.ChAStatusXferToString

Example

```
Sub Main
   Dim String_Array(5)
   String_Array(0)= "Not Indicated"
   String_Array(1)= "20 bits"
   String_Array(2)= "19 bits"
   String_Array(3)= "18 bits"
   String_Array(4)= "17 bits"
```

```
   String_Array(5)= “16 bits”

   System = AP.Application.SysType
   AP.Application.NewTest  ‘Reset panels
   AP.Bits.Mode = 1     ‘Professional Mode
   If System = “1" Then
      AP.S1Dio.InFormat = 2
   Else
      AP.S2Dio.InFormat = 3
   End If
   Wait .5  ‘Wait for reading to update
   Debug.Print “Word Length Reading = ” & _
      String_Array(AP.Bits.ChAWordLengthRdg)
End Sub
```

Example Output

```
Word Length Reading = Not Indicated
```

AP.Bits.ChAXmitData

❶❷ Property

Obsolete Obsolete command not recomended for new design.

Syntax **AP.Bits.ChAXmitData(*string$*)**

Data Type Integer

Parameters

Part	Description
string$	Array containing status bit information.

Description This command transmits the status bits data contained in the string for channel A.

Note: This command has been replaced by the `AP.Bits.ChAXmitStatus` command. The `AP.Bits.ChAXmitData` command will continue to function as documented here but will be removed from all visible areas within APWIN.

AP.Bits.ChAXmitStatus ❶❷ Property

Syntax **AP.Bits.ChAXmitStatus(*string$*)**

Data Type Integer

Parameters

Part	Description
string$	Array containing status bit information.

Description This command transmits the status bits data contained in the string for channel A.

See Also AP.Bits.ChAStatusXferToString

Example

```
Sub Main
   With AP.Bits
   'Get current Channel A&B status
      Channel_A_Status = .ChAStatusXferToString
      Channel_B_Status = .ChBStatusXferToString

      'Your code goes here

   'Restore Channel A&B status
      .ChAXmitStatus = Channel_A_Status
      .ChBXmitStatus = Channel_B_Status
   End With
End Sub
```

AP.Bits.ChBAudioModeRdg ❷ Property

Syntax **AP.Bits.ChBAudioModeRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
1	Normal

0	Non Audio

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAAudioModeRdg`.

AP.Bits.ChBAuxBitsRdg ❷ Property

Syntax **`AP.Bits.ChBAuxBitsRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	20-bit not defined
1	24-bit not defined
2	20-bit single
3	Reserved

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAAuxBitsRdg`.

AP.Bits.ChBCategoryRdg ❷ Property

Syntax `AP.Bits.ChBCategoryRdg([string$])`

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	General
1	CD Player
2	PCM Adaptor
3	DAT Recorder
4	Digital Broadcast
5	Musical Instrument

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChACategoryRdg`.

AP.Bits.ChBChModeRdg ❷ Property

Syntax `AP.Bits.ChBChModeRdg([string$])`

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description

0	Not Indicated
1	2-channel
2	Single-channal
3	Primary/Sec
4	Stereo
5	Reserved-1
6	Reserved-2
7	Vector to byte 3

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAChModeRdg`.

AP.Bits.ChBChNumRdg ❷ Property

Syntax **`AP.Bits.ChBChNumRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Don't Care
1	A (Left)
2	B (Right)
3	C
4	D
5	E
6	F
7	G

8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAChNumRdg`.

AP.Bits.ChBClockAccuracyRdg ❷ Property

Syntax **`AP.Bits.ChBClockAccuracyRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Level 1
1	Level 2
2	Level 3
3	Reserved

Description This command returns the Status Bits channel B Clock Accuracy 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAClockAccuracyRdg`.

AP.Bits.ChBCopyrightRdg ❷ Property

Syntax **`AP.Bits.ChBCopyrightRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
`string$`	Optional string containing status bit information.

Result

Value	Description
1	Copyright
0	Non-Copyright

Description This command returns the Status Bits channel B Copyright 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChACopyrightRdg`.

AP.Bits.ChBCrcRdg ❷ Property

Syntax **`AP.Bits.ChBCrcRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
`string$`	Optional string containing status bit information.

Result

Value	Description
0	Invalid
1	Valid

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChACrcRdg`.

AP.Bits.ChBDestinationRdg ❷ Property

Syntax **`AP.Bits.ChBDestinationRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChADestinationRdg`.

AP.Bits.ChBEmphRdg ❷ Property

Syntax **`AP.Bits.ChBEmphRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
The following list is for Consumer Mode.	
0	No Pre-emph
1	50/15S
The following list is for Professional Mode.	
0	Not Indicated
1	None
2	50/15 uS
3	CCITT J.17

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAEmphRdg`.

AP.Bits.ChBFlag0_5Rdg ❷ Property

Syntax **`AP.Bits.ChBFlag0_5Rdg([`*`string$`*`])`**

Data Type Boolean

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
True	Set

False	Clear

Description This command returns the Status Bits channel B Flag 0-5 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAFlag0_5Rdg`.

AP.Bits.ChBFlag6_13Rdg ❷ Property

Syntax **`AP.Bits.ChBFlag6_13Rdg([`*`string$`*`])`**

Data Type Boolean

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
True	Set
False	Clear

Description This command returns the Status Bits channel B Flag 6-13 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAFlag0_5Rdg`.

AP.Bits.ChBFlag14_17Rdg ❷ Property

Syntax **`AP.Bits.ChBFlag14_17Rdg([`*`string$`*`])`**

Data Type Boolean

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
True	Set
False	Clear

Description This command returns the Status Bits channel B Flag 14-17 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAFlag0_5Rdg`.

AP.Bits.ChBFlag18_21Rdg ❷ Property

Syntax **`AP.Bits.ChBFlag18_21Rdg([`*`string$`*`])`**

Data Type Boolean

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
True	Set
False	Clear

Description This command returns the Status Bits channel B Flag 18-21 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 `AP.Bits.ChBStatusXferToString` command.

See Also AP.Bits.ChBStatusXferToString

Example See example for AP.Bits.ChAFlag0_5Rdg.

AP.Bits.ChBFreqModeRdg ❷ Property

Syntax **AP.Bits.ChBFreqModeRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Unlocked
1	Locked

Description This command returns the Status Bits channel B 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 AP.Bits.ChBStatusXferToString command.

See Also AP.Bits.ChBStatusXferToString

Example See example for AP.Bits.ChAFreqModeRdg.

AP.Bits.ChBLocalAddressRdg ❷ Property

Syntax **AP.Bits.ChBLocalAddressRdg([*string$*])**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChALocalAddressRdg`.

AP.Bits.ChBModeRdg ❷ Property

Syntax **`AP.Bits.ChBModeRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Consumer
1	Professional

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAModeRdg`.

AP.Bits.ChBOriginRdg ❷ Property

Syntax **`AP.Bits.ChBOriginRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAOriginRdg`.

AP.Bits.ChBRefSignalRdg ❷ Property

Syntax **`AP.Bits.ChBRefSignalRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Nor a ref. Signal
1	Grade 1
2	Grade 2
3	Reserved

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChARefSignalRdg`.

AP.Bits.ChBSampleFreqRdg ❷ Property

Syntax `AP.Bits.ChBSampleFreqRdg([string$])`

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	48 kHz
1	44.1 kHz
2	32 kHz

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChASampleFreqRdg`.

AP.Bits.ChBSourceNumRdg ❷ Property

Syntax `AP.Bits.ChBSourceNumRdg([string$])`

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	Don't Care
1	1
2	2

3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChASourceNumRdg`.

AP.Bits.ChBStatusXferToArray ❷ Property

Obsolete Obsolete command not recomended for new design.

Syntax **`AP.Bits.ChBStatusXferToArray`**

Data Type Variant

Parameters None

Description This command transfers the contents of the channel B Status to an string.

Note: This command has been replaced by the `AP.Bits.ChBStatusXferToString` command. The

`AP.Bits.ChBStatusXferToArray` command will continue to function as documented here but will be removed from all visible areas within APWIN.

AP.Bits.ChBStatusXferToString ❷ Property

Syntax **`AP.Bits.ChBStatusXferToString`**

Data Type Variant

Parameters None

Description This command transfers the contents of the channel B Status to an string.

See Also `AP.Bits.ChBXmitData`

Example See example for `AP.Bits.ChAStatusXferToString`.

AP.Bits.ChBTimeOfDayRdg ❷ Property

Syntax **`AP.Bits.ChBTimeOfDayRdg([`*`string$`*`])`**

Data Type Integer

Parameters

Part	Description
`string$`	Optional string containing status bit information.

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChATimeOfDayRdg`.

AP.Bits.ChBUserBitsRdg ❷ Property

Syntax `AP.Bits.ChBUserBitsRdg([string$])`

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description
0	None
1	192-bit block
2	Reserved
3	User defined

Description This command returns the Status Bits channel B 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAUserBitsRdg`.

AP.Bits.ChBWordLengthRdg ❷ Property

Syntax `AP.Bits.ChBWordLengthRdg([string$])`

Data Type Integer

Parameters

Part	Description
string$	Optional string containing status bit information.

Result

Value	Description

0	Not Indicated
1	20 bits
2	19 bits
3	18 bits
4	17 bits
5	16 bits

Description This command returns the Status Bits channel B Word Length 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 `AP.Bits.ChBStatusXferToString` command.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAWordLengthRdg`.

AP.Bits.ChBXmitData ❷ Property

Obsolete Obsolete command not recomended for new design.

Syntax **`AP.Bits.ChBXmitData(`*`string$`*`)`**

Data Type Integer

Parameters

Part	Description
`string$`	Array containing status bit information.

Description This command transmits the status bits data contained in the string for channel B.

Note: This command has been replaced by the `AP.Bits.ChBXmitStatus` command. The `AP.Bits.ChBXmitData` command will continue to function as documented here but will be removed from all visible areas within APWIN.

AP.Bits.ChBXmitStatus ❷ Property

Syntax `AP.Bits.ChBXmitStatus(string$)`

Data Type Integer

Parameters

Part	Description
string$	Array containing status bit information.

Description This command transmits the status bits data contained in the string for channel B.

See Also `AP.Bits.ChBStatusXferToString`

Example See example for `AP.Bits.ChAXmitStatus`.

AP.Bits.Cons.AudioMode ❶❷ Property

Syntax `AP.Bits.Cons.AudioMode`

Parameters None

Data Type Integer

1	Audio Mode
0	Data Mode

Description This command sets the Mode parameter encoded in the Consumer Status Bits.

Example See example for `AP.Bits.Cons.Category`.

AP.Bits.Cons.Category ❶❷ Property

Syntax `AP.Bits.Cons.Category`

Parameters None

Data Type Integer

0	General

1	CD Player
2	PCM Adaptor
3	DAT Recorder
4	Digital Broadcast
5	Musical Instrument

Description This command sets the Catagory 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 ...
   AP.Bits.XmitChannel = 2        'channels A & B
   AP.Bits.Mode = 0               'consumer
   AP.Bits.Cons.AudioMode = 1     'data mode
   AP.Bits.Cons.CopyRight = 1     'non-copyright
   AP.Bits.Cons.Emphasis = 1      '50/15 uS
   AP.Bits.Cons.Channels = 0      '2 channel
   AP.Bits.Cons.Category = 1      'CD player
   AP.Bits.Cons.SourceNum = 1     'source 1
   AP.Bits.Cons.ChNum = 1         'A (left)
   AP.Bits.Cons.SampleFreq = 0    '48 kHz
   AP.Bits.Cons.ClockAccuracy = 0 'level 1
   'rest of program ...
End Sub
```

AP.Bits.Cons.Channels

02 Property

Syntax `AP.Bits.Cons.Channels`

Parameters None

Data Type Integer

1	2 Channel
0	4 Channel

Description This command sets the Channel Mode parameter encoded in the Consumer Status Bits.

See Also AP.

Example See example for AP.Bits.Cons.Category.

AP.Bits.Cons.ChNum ❶❷ Property

Syntax **AP.Bits.Cons.ChNum**

Parameters None

Data Type Integer

0	Don't Care
1	A (Left)
2	B (Right)
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O

Description This command sets the Source Number parameter encoded in the Consumer Status Bits.

See Also AP.Bits.XmitChannel

Example See example for AP.Bits.Cons.Category.

AP.Bits.Cons.ClockAccuracy

02 Property

Syntax `AP.Bits.Cons.ClockAccuracy`

Parameters None

Data Type Integer

0	Level 1
1	Level 2
2	Level 3
3	Reserved

Description This command sets the Clock Accuracy parameter encoded in the Consumer Status Bits.

Example See example for `AP.Bits.Cons.Category`.

AP.Bits.Cons.Copyright

02 Property

Syntax `AP.Bits.Cons.Copyright`

Parameters None

Data Type Integer

1	Copyright
0	Non-Copyright

Description This command sets the Copyright parameter encoded in the Consumer Status Bits.

Example See example for `AP.Bits.Cons.Category`.

AP.Bits.Cons.Emphasis

02 Property

Syntax `AP.Bits.Cons.Emphasis`

Parameters None

Data Type Integer

1	No Pre-emph
0	50/15S

Description This command sets the Emphasis parameter encoded in the Consumer Status Bits.

Example See example for `AP.Bits.Cons.Category`.

AP.Bits.Cons.SampleFreq ❶❷ Property

Syntax `AP.Bits.Cons.SampleFreq`

Parameters None

Data Type Integer

0	48 kHz
1	44.1 kHz
2	32 kHz

Description This command sets the Frequency parameter encoded in the Consumer Status Bits.

Example See example for `AP.Bits.Cons.Category`.

AP.Bits.Cons.SourceNum ❶❷ Property

Syntax `AP.Bits.Cons.SourceNum`

Parameters None

Data Type Integer

0	Don't Care
1	1
2	2
3	3
4	4
5	5
6	6

7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16

Description This command sets the Source Number parameter encoded in the Consumer Status Bits.

Example See example for `AP.Bits.Cons.Category`.

AP.Bits.Mode

01 02 Property

Syntax `AP.Bits.Mode([string$])`

Parameters None

Data Type Integer

0	Consumer
1	Professional

Description This command sets the Transmit Mode.

Example See example for `AP.Bits.Cons.Category`.

AP.Bits.Pro.AudioMode

01 02 Property

Syntax `AP.Bits.Pro.AudioMode`

Parameters None

Data Type Integer

0	Normal

1	Non Audio

Description This command sets the Audio Mode parameter encoded in the Professional Status Bits.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.AuxBits

❶❷ Property

Syntax `AP.Bits.Pro.AuxBits`

Parameters None

Result Integer

0	20-bit not defined
1	24-bit main audio
2	20-bit single
3	Reserved

Description This command sets the Aux Bits parameter encoded in the Professional Status Bits.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.ChMode

❶❷ Property

Syntax `AP.Bits.Pro.ChMode`

Parameters None

Result Integer

0	Not Indicated
1	2-channel
2	Single-channal
3	Primary/Sec
4	Stereo

5	Reserved-1
6	Reserved-2
7	Vector to byte 3

Description This command sets the Channel Mode parameter encoded in the Professional Status Bits.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.CrcEnable ❷ Property

Syntax **`AP.Bits.Pro.CrcEnable`**

Parameters None

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 `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.Destination ❶❷ Property

Syntax **`AP.Bits.Pro.Destination`**

Data Type String

Parameters None

Description This command sets a four-character alphanumeric (ASCII) code to be transmitted.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.Emphasis ❶❷ Property

Syntax `AP.Bits.Pro.Emphasis`

Parameters None

Result Integer

0	Not Indicated
1	None
2	50/15 uS
3	CCITT J.17

Description This command sets the Emphasia parameter encoded in the Professional Status Bits.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.Flag0_5 ❶❷ Property

Syntax `AP.Bits.Pro.Flag0_5`

Parameters None

Result Boolean

True	Set
False	Clear

Description This command sets or clears the Reliability Flag for bytes 0-5.

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.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.Flag6_13

01 02 Property

Syntax `AP.Bits.Pro.Flag6_13`

Parameters None

Result Boolean

True	Set
False	Clear

Description This command sets or clears the Reliability Flag for bytes 6-13.

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 eight status bytes is valid.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.Flag14_17

01 02 Property

Syntax `AP.Bits.Pro.Flag14_17`

Parameters None

Result Boolean

True	Set
False	Clear

Description This command sets or clears the Reliability Flag for bytes 14-17.

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 four status bytes is valid.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.Flag18_21 ❶❷ Property

Syntax **`AP.Bits.Pro.Flag18_21`**

Parameters None

Result Boolean

`True`	Set
`False`	Clear

Description This command sets or clears the Reliability Flag for bytes 18-21.

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 four status bytes is valid.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.FreqMode ❶❷ Property

Syntax **`AP.Bits.Pro.FreqMode`**

Parameters None

Result Integer

`0`	Unlocked
`1`	Locked

Description This command sets the Frequency Mode parameter encoded in the Professional Status Bits.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.LocalAddress ❷ Property

Syntax **`AP.Bits.Pro.LocalAddress`**

Data Type Double

Parameters None

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 `AP.Bits.Pro.AddressAuto`

Example

```
Sub Main
   'other setup code ...
   AP.Bits.XmitChannel = 1       'transmit chan B
   AP.Bits.Mode = 1              'professional
   AP.Bits.Pro.AudioMode = 1     'non-audio
   AP.Bits.Pro.Emphasis = 2      '50/15 uS
   AP.Bits.Pro.FreqMode = 0      'unlocked
   AP.Bits.Pro.SampleFreq = 1    '48 kHz
   AP.Bits.Pro.ChMode = 4        'stereo
   AP.Bits.Pro.UserBits = 3      'user defined
   AP.Bits.Pro.AuxBits = 1       '24-bit main audio
   AP.Bits.Pro.WordLength = 1    '24 bits
   AP.Bits.Pro.RefSignal = 2     'grade 2
   AP.Bits.Pro.Origin = "SYS2"      'source SYS2
   AP.Bits.Pro.Destination = "TEST" 'target TEST
   AP.Bits.Pro.LocalAddressAuto = 0    'auto address off
   AP.Bits.Pro.LocalAddress = 123456   'set address = _
      123456
   AP.Bits.Pro.TimeOfDay = 1234  'set TOD = 1234 samples
   AP.Bits.Pro.Flag0_5 = True    'unreliable
   AP.Bits.Pro.Flag6_13 = True      'unreliable
   AP.Bits.Pro.Flag14_17 = False 'reliable
   AP.Bits.Pro.Flag18_21 = True  'unreliable
```

```
    AP.Bits.Pro.CrcEnable = True  'valid
    'Rest of program
End Sub
```

AP.Bits.Pro.LocalAddressAuto ❷ Property

Syntax **AP.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 enered into this field via the `AP.Bits.Pro.LocalAddress` command and the number will be continuously transmitted as the Local Address code in the status bytes.

See Also `AP.Bits.Pro.LocalAddress, AP.Bits.Pro.TimeOfDay`

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.Origin ❶❷ Property

Syntax **AP.Bits.Pro.Origin**

Data Type String

Parameters

Name	Description
unit$	The following units are available:

Description This command sets a four-character alphanumeric (ASCII) code to be transmitted.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.RefSignal

01 02 Property

Syntax `AP.Bits.Pro.RefSignal`

Parameters None

Result Integer

0	Nor a ref. Signal
1	Grade 1
2	Grade 2
3	Reserved

Description This command sets the ReferenceSignal parameter encoded in the Professional Status Bits.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.SampleFreq

01 02 Property

Syntax `AP.Bits.Pro.SampleFreq`

Parameters None

Result Integer

0	Not Indicated
1	48 kHz
2	44.1 kHz
3	32 kHz

Description This command sets the Frequency parameter encoded in the Professional Status Bits.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.TimeOfDay ❶❷ Property

Syntax `AP.Bits.Pro.TimeOfDay`

Data Type Double

Parameters None

Description This command sets the Time Of Day parameter encoded in the Professional Status Bits bytes 18-21.

See Also `AP.Bits.Pro.AddressAuto`

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.UserBits ❶❷ Property

Syntax `AP.Bits.Pro.UserBits`

Parameters None

Result Integer

0	None
1	192-bit block
2	Reserved
3	User defined

Description This command sets the User Bits parameter encoded in the Professional Status Bits.

Example See example for `AP.Bits.Pro.LocalAddress`.

AP.Bits.Pro.WordLength ❶❷ Property

Syntax `AP.Bits.Pro.WordLength`

Parameters None

Result Integer

The following list contains the selections relevant to the AP.Bits.Pro.AuxBits command "20-bit not defined" selection.

0	Not Indicated
1	20 bits
2	19 bits
3	18 bits
4	17 bits
5	16 bits

The following list contains the selections relevant to the AP.Bits.Pro.AuxBits command "24-bit main audio" selection.

0	Not Indicated
1	24 bits
2	23 bits
3	22 bits
4	21 bits
5	20 bits

Description This command sets the Audio Word Length parameter encoded in the Professional Status Bits.

Example See example for AP.Bits.Pro.LocalAddress.

AP.Bits.XmitChannel ❷ Method

Syntax **AP.Bits.XmitChannel**

Parameters None

Result Integer

0	A
1	B
2	A & B

Description This command sets the Transmit Channel.

Example See example for AP.Bits.Pro.LocalAddress.

User Notes

User Notes

RS-232

AP.CommA.Break ❶❷ Property

AP.CommB.Break ❶❷ Property

Syntax `AP.CommA.Break`

Parameters None

Data Type Boolean

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

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.

AP.CommA.CDHolding ❶❷ Property

AP.CommB.CDHolding ❶❷ Property

Syntax `AP.CommA.CDHolding`

Parameters None

Data Type Boolean

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

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 AP.CommA.CDTimeout command has expired, the AP.CommA.CommEvent command is set to comCDTO (Carrier Detect Timeout Error), and a OnComm event is generated.

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

See Also `AP.CommA.CDTimeout`

AP.CommA.CDTimeout 02 Property

AP.CommB.CDTimeout 02 Property

Syntax **`AP.CommA.CDTimeout`**

Parameters None

Data Type Long

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 AP.CommA.CommEvent command to CDTO (Carrier Detect Timeout Error) and generating the OnComm event.

AP.CommA.CommEvent 02 Property

AP.CommB.CommEvent 02 Property

Syntax *`Event = AP.CommA.CommEvent`*

Parameters None

Result Integer

The following list contains communications errors or events.

Setting	Value	Description
`comBreak`	`1001`	Break signal received.

comCTSTO	*1002*	Clear To Send Timeout. The Clear To Send line was low for the number of milliseconds specified by the AP.CommA.CTSTimeout command while trying to send a character.
comDSRTO	*1003*	Data Set Ready Timeout. The Data Set Ready line was low for the number of milliseconds specified by the AP.CommA.DSRTimeout command while trying to send a character.
comFrame	*1004*	Framing Error. The hardware detected a framing error.
comOverrun	*1006*	Port Overrun. A character was not read from the hardware before the next character arrived and was lost.
comCDTO	*1007*	Carrier Detect Timeout. The Carrier Detect line was low for the number of milliseconds specified by the AP.CommA.CDTimeout command while trying to send a character.
comRxOver	*1008*	Receive Buffer Overflow. The receive buffer is full.
comRxParity	*1009*	Parity Error. Parity error detected.
comTxFull	*1010*	Transmit Buffer Full. The transmit buffer was full while trying to queue a character.

Communications events include the following settings.

Setting	Value	Description
comEvSend	*1*	There are fewer than SThreshold number of characters in the transmit buffer.
comEvReceive	*2*	Received RThreshold number of characters. This event is generated continuously until you use the Input property to remove the data from the receive buffer.
comEvCTS	*3*	Change in Clear To Send line.
comEvDSR	*4*	Change in Data Set Ready line. This event is only fired when DSR changes from 1 to 0.
comEvDC	*5*	Change in Carrier Detect line.
comEvRing	*6*	Ring detected. Some UARTs (universal asynchronous receiver-transmitters) may not support this event.
comEvEOF	*7*	End Of File (ASCII character 26) character received.

Description Returns the most recent communication event or error.

AP.CommA.CommId 12 Property

AP.CommB.CommId 12 Property

Syntax `AP.CommA.CommICommId`

Parameters None

Result Integer

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

AP.CommA.CommPort 12 Property

AP.CommB.CommPort 12 Property

Syntax `AP.CommA.CommPort`

Data Type Integer

Description his 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 AP.CommA.CommPort before opening the port.

Example

```
Sub Main
   If AP.CommA.PortOpen = True Then 'Close Port if Open
      AP.CommA.PortOpen = False
   End If

   'Port Setup
   AP.CommA.CommPort = 2         'Select Comm Port
   AP.CommA.Settings = "9600,N,8,1" 'Set Comm Port _
      settings baud rate etc.
   AP.CommA.OutBufferSize = 10 'Set Output buffer size
   AP.CommA.InBufferSize = 10 'Set Input buffer size

   'Output to Comm Port 2
```

```
   AP.CommA.PortOpen = True          'Open Comm Port 2
   AP.CommA.Output = "1234567890"    'Send data

   'Input from Comm Port 2
   Character$ = AP.CommA.Input 'Get data sent to Comm 2
   Debug.Print Character$     'Print Input to _
      Immediate Window

   AP.CommA.PortOpen = False         'Close Comm Port
End Sub
```

AP.CommA.CTSHolding ❶❷ Property

AP.CommB.CTSHolding ❶❷ Property

Syntax `AP.CommA.CTSHolding`

Parameters None

Data Type Boolean

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

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 AP.CommA.CTSTimeout command has expired, the AP.CommA.CommEvent command is set to comCTSTO (Clear To Send Timeout) and a OnComm event is generated.

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

See Also `AP.CommA.Handshaking`

AP.CommA.CTSTimeout 12 Property

AP.CommB.CTSTimeout 12 Property

Syntax `AP.CommA.CTSTimeout`

Parameters None

Data Type Long

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 AP.CommA.CommEvent command to CTSTO (Clear To Send Timeout Error) and generating the OnComm event.

AP.CommA.DSRHolding 12 Property

AP.CommB.DSRHolding 12 Property

Syntax `AP.CommA.DSRHolding`

Parameters None

Data Type Boolean

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

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.

AP.CommA.DSRTimeout 12 Property

AP.CommB.DSRTimeout 12 Property

Syntax `AP.CommA.DSRTimeout`

Parameters None

Data Type Long

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 AP.CommA.CommEvent command to DSRTO (Data Set Ready Timeout Error) and generating the OnComm event.

AP.CommA.DTREnable ❶❷ Property

AP.CommB.DTREnable ❶❷ Property

Syntax `AP.CommA.DTREnable`

Parameters None

Data Type Boolean

True	Enable the Data Terminal Ready (line high) when port opened and (line Low) when the port is closed.
False	(Default) Disable the Data Terminal Ready (line always low).

Description 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.

Setting the Data Terminal Ready line to low in most cases hangs up the telephone.

AP.CommA.Handshaking ❶❷ Property

AP.CommB.Handshaking ❶❷ Property

Syntax `AP.CommA.Handshaking`

Parameters None

Data Type Integer

Valid protocols are listed in the following table.

0	(Default) No handshaking.
1	XON/XOFF handshaking.
2	RTS/CTS (Request To Send/Clear To Send) handshaking.
3	Both Request To Send and XON/XOFF handshaking.

Description 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.

AP.CommA.InBufferCount ❶❷ Property

AP.CommB.InBufferCount ❶❷ Property

Syntax `AP.CommA.InBufferCount`

Parameters None

Data Type Integer

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

AP.CommA.InBufferSize ❶❷ Property

AP.CommB.InBufferSize ❶❷ Property

Syntax `AP.CommA.InBufferSize`

Parameters None

Data Type Integer

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

Example See example for `AP.CommA.CommPort.`

AP.CommA.Input ❶❷ Property

AP.CommB.Input ❶❷ Property

Syntax **`AP.CommA.Input`**

Parameters None

Result String

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

Example See example for `AP.CommA.CommPort.`

AP.CommA.InputLen ❶❷ Property

AP.CommB.InputLen ❶❷ Property

Syntax **`AP.CommA.InputLen`**

Parameters None

Data Type Integer

Description This command sets and returns the number of characters the AP.CommA.Input command reads from the receive buffer.

Setting the AP.CommA.InputLen command to 0 causes the AP.CommA.Input command to read the entire contents of the receive buffer.

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

AP.CommA.Interval 12 Property

AP.CommB.Interval 12 Property

Syntax `AP.CommA.Interval`

Parameters None

Data Type Long

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

AP.CommA.NullDiscard 12 Property

AP.CommB.NullDiscard 12 Property

Syntax `AP.CommA.NullDiscard`

Parameters None

Data Type Boolean

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

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

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

AP.CommA.OutBufferCount ❶❷ Property

AP.CommB.OutBufferCount ❶❷ Property

Syntax `AP.CommA.OutBufferCount`

Parameters None

Data Type Integer

Description This command returns the number of characters in the transmit buffer. The transmit buffer can be cleared by setting the AP.CommA.OutBufferCount command to 0.

AP.CommA.OutBufferSize ❶❷ Property

AP.CommB.OutBufferSize ❶❷ Property

Syntax `AP.CommA.OutBufferSize`

Parameters None

Data Type Integer

Description This command sets and returns the size, in characters, of the transmit buffer. The default transmit buffer size is 512 bytes.

Example See example for `AP.CommA.CommPort`.

AP.CommA.Output ❶❷ Property

AP.CommB.Output ❶❷ Property

Syntax `AP.CommA.Output`

Parameters None

Data Type String

Description This command sends a string of characters to the transmit buffer.

Example See example for `AP.CommA.CommPort`.

AP.CommA.ParityReplace 12 Property

AP.CommB.ParityReplace 12 Property

Syntax **`AP.CommA.ParityReplace`**

Parameters None

Data Type String

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.

AP.CommA.PortOpen 12 Property

AP.CommB.PortOpen 12 Property

Syntax **`AP.CommA.PortOpen`**

True | False =

Parameters None

Data Type Boolean

`True`	Port is opened.
`False`	Port is closed or closes the port and clears the receive and transmit buffers.

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

If either the AP.CommA.DTREnable or the AP.CommA.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 `AP.CommA.CommPort.`

AP.CommA.RThreshold ❶❷ Property

AP.CommB.RThreshold ❶❷ Property

Syntax **`AP.CommA.RThreshold`**

Parameters None

Data Type Integer

Description This command sets and returns the number of characters to receive before the communications control sets the CommEvent command to comEvReceive and generates the OnComm event.

By setting the AP.CommA.RThreshold command to 0 (the default) generationof the OnComm event is disabled when characters are received.

By setting AP.CommA.RThreshold command to 1, each time a character is placed in the receive buffer an OnComm event is generated.

AP.CommA.RTSEnable ❶❷ Property

AP.CommB.RTSEnable ❶❷ Property

Syntax **`AP.CommA.RTSEnable`**

Parameters None

Data Type Boolean

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

Description This command determines the state of the Request To Send line.

The Request To Send line is used in RTS/CTS hardware handshaking.

AP.CommA.Settings ❶❷ Property

AP.CommB.Settings ❶❷ Property

Syntax `AP.CommA.Settings`

Parameters None

Data Type String

The following table lists the valid baud rates.

Setting	Description
110	
300	
600	
1200	
2400	
9600	(Default)
14400	
19200	

The following table describes the valid parity values.

Setting	Description
E	Even
M	Mark
N	None (Default)
O	Odd
S	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	

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
B	Baud rate
P	Parity
D	Number of data bits
S	Number of stop bits

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

Example See example for `AP.CommA.CommPort.`

AP.CommA.SThreshold ❶❷ Property

AP.CommB.SThreshold ❶❷ Property

Syntax `AP.CommA.SThreshold`

Parameters None

Data Type Integer — Valid protocols are listed in the following table.

0	No handshaking.
1	XON/XOFF handshaking.
2	RTS/CTS (Request To Send/Clear To Send) handshaking.
3	Both Request To Send and XON/XOFF handshaking.

Description This command sets and returns the minimum number of characters allowed in the transmit buffer before the communications control sets the CommEvent property to comEvSend.

Setting the AP.CommA.SThreshold command to 0 (the default) disables generating the OnComm event for data transmission events. Setting the AP.CommA.SThreshold command to 1 causes the communications control to generate the OnComm event when the transmit buffer is completely empty.

If the number of characters in the transmit buffer is less than the number specified by the AP.CommA.SThreshold command, the CommEvent property is set to comEvSend. The comEvSend event is only sent once, when the number of characters crosses the Threshold.

User Notes

User Notes

User Notes

User Notes

User Notes

Computes

AP.Compute.Avg.Apply ❶❷ Method

Syntax **AP.Compute.Avg.Apply**

Parameters None

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 AP.Compute.Clear.All, AP.Compute.Avg.Data, AP.Compute.Avg.PostSweep, AP.Compute.Avg.Start, AP.Compute.Avg.Stop,

❶ **Example**

```
Sub Main
   AP.File.OpenTest "AVG1.AT2"          'Open test
   AP.Compute.Clear.All
   AP.Compute.Avg.Data(1) = True 'Use Column1 for data 1
   AP.Compute.Avg.PostSweep = False     'Disable Apply _
      after sweep
   AP.Compute.Avg.Start("Hz") = 5000
   AP.Compute.Avg.Stop("Hz") = 100
   AP.Sweep.Start
   AP.Compute.Avg.Apply           'Compute Average
End Sub
```

AP.Compute.Avg.Data ❶❷ Property

Syntax **AP.Compute.Avg.Data(***column%***)**

Data Type Boolean

Parameters

Name	Description
column%	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 columns, several Average computations can be performed in one operation.

See Also `AP.Compute.Avg.Apply`

Example See example for `AP.Compute.Avg.Apply`.

AP.Compute.Avg.PostSweep ❶❷ Property

Syntax **`AP.Compute.Avg.PostSweep`**

Parameters None

Data Type Boolean

True	Enable computation to be applyed 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.

APWIN 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 `AP.Compute.Avg.Apply`

Example See example for `AP.Compute.Avg.Apply`.

AP.Compute.Avg.Start ❶❷ Property

Syntax **AP.Compute.Avg.Start(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	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 AP.Compute.Avg.Stop, AP.Compute.Avg.Apply

Example See example for AP.Compute.Avg.Apply.

AP.Compute.Avg.Stop ❶❷ Property

Syntax **AP.Compute.Avg.Stop(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	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 AP.Compute.Avg.Start, AP.Compute.Avg.Apply

Example See example for AP.Compute.Avg.Apply.

AP.Compute.Center.Apply ❶❷ Method

Syntax **AP.Compute.Center.Apply**

Parameters None

Result Boolean

True	Computation performed.
False	Computation NOT performed.

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

See Also `AP.Compute.Clear.All, AP.Compute.Center.Data, AP.Compute.Center.PostSweep, AP.Compute.Center.Start, AP.Compute.Center.Stop`

❷ Example

```
Sub Main
   Dim Status As Boolean
   Status = AP.Log.Enable          'Determine Log Status
   AP.Log.Enable = False           'Enable Log Status
   AP.File.OpenTest "CENTER1.AT2" 'Open test
   AP.Compute.Clear.All
   AP.Compute.Center.Data(1) = True 'Set Data 1
   AP.Compute.Center.PostSweep = False 'Apply after _
      sweep off
   AP.Compute.Center.Start("Hz") = 200000 'Start freq _
      200k Hz
   AP.Compute.Center.Stop("Hz") = 10 'Stop at 10 Hz
   AP.Sweep.Start
   AP.Compute.Center.Apply         'Compute Center
   AP.Log.Enable = Status          'Reset Log Status
End Sub
```

AP.Compute.Center.Data

❶❷ Property

Syntax **AP.Compute.Center.Data(***column%***)**

Data Type Boolean

Parameters

Name	Description
column%	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 columns, several Center computations can be performed in one operation.

See Also `AP.Compute.Center.Apply`

Example See example for `AP.Compute.Center.Apply`.

AP.Compute.Center.PostSweep ❶❷ Property

Syntax **`AP.Compute.Center.PostSweep`**

Parameters None

Data Type Boolean

`True`	Enable computation to be applyed 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.

APWIN 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 `AP.Compute.Center.Apply`

Example See example for `AP.Compute.Center.Apply`.

AP.Compute.Center.Start ❶❷ Property

Syntax **`AP.Compute.Center.Start(`***`unit$`***`)`**

Data Type Double

Parameters

Name	Description

	unit$	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 Center computation will be performed.

See Also `AP.Compute.Center.Stop, AP.Compute.Center.Apply`

Example See example for `AP.Compute.Center.Apply`.

AP.Compute.Center.Stop

❶❷ Property

Syntax **`AP.Compute.Center.Stop(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
unit$	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 Center computation will be performed.

See Also `AP.Compute.Center.Start, AP.Compute.Center.Apply`

Example See example for `AP.Compute.Center.Apply`.

AP.Compute.Clear.All

❶❷ Method

Syntax **`AP.Compute.Clear.All`**

Data Type Void

Parameters None

Result Void

Description This command clears all computes from the current test.

Example See example for `AP.Compute.Center.Apply`.

AP.Compute.Delta.Apply ❶❷ Method

Syntax **AP.Compute.Delta.Apply**

Parameters None

Result Boolean

True	Computation performed.
False	Computation NOT performed.

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

See Also AP.Compute.Clear.All, AP.Compute.Delta.Data, AP.Compute.Delta.FileName, AP.Compute.Delta.PostSweep

❷ **Example**

```
Sub Main
   AP.File.OpenTest "DELTA1.AT2" 'Opens test to be run _
      with results compared to stored data file
   AP.Compute.Clear.All
   AP.Sweep.Start
   AP.Compute.Delta.FileName = "DELTA1.ADA" 'Data file _
      used in delta computation
   AP.Compute.Delta.PostSweep = False 'Disables apply _
      after sweep
   AP.Compute.Delta.Data(1,1) = True
   AP.Compute.Delta.Data(2,2) = True
   AP.Compute.Delta.Apply  'Compute Delta
End Sub
```

AP.Compute.Delta.Data ❶❷ Property

Syntax **AP.Compute.Delta.Data(***source%, column%***)**

Data Type Boolean

Parameters

Name	Description
source%	Number of the Sweep Data (1-6) of the data in memory.

column%	Number of the Data Column (0-7) of the data specified by the AP.Compute.Delta.FileName command.

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

See Also AP.Compute.Delta.Apply, AP.Compute.Delta.FileName

Example See example for AP.Compute.Delta.Apply.

AP.Compute.Delta.FileName

12 Property

Syntax **AP.Compute.Delta.FileName(*filename$*)**

Parameters

Name	Description
filename$	Any valid DOS filename and extension. Enter "SweepData" for the file name to select data in memory.

Result Boolean

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 AP.Compute.Delta.Apply

Example See example for AP.Compute.Delta.Apply.

AP.Compute.Delta.PostSweep

12 Property

Syntax **AP.Compute.Delta.PostSweep**

Parameters None

Data Type Boolean

True	Enable computation to be applyed 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.

APWIN 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 `AP.Compute.Delta.Apply`

Example See example for `AP.Compute.Delta.Apply`.

AP.Compute.Equalize.Apply ❶❷ Method

Syntax **`AP.Compute.Equalize.Apply`**

Parameters None

Result Boolean

True	Computation performed.
False	Computation NOT performed.

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

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

❷ **Example**

```
Sub Main
   AP.File.OpenTest “Eq1.at2"    ‘opens test to be run
      ‘with results compared to stored data file
   AP.Compute.Clear.All
   AP.Sweep.Start
   AP.Compute.Equalize.FileName = “Eq1.ada”  ‘data
      ‘file used in delta computation
   AP.Compute.Equalize.PostSweep = False ‘disables
      ‘apply after sweep
```

```
    AP.Compute.Equalize.Data(1,1) = True
    AP.Compute.Equalize.Data(2,2) = True
    AP.Compute.Equalize.Apply  'Compute Equalize
End Sub
```

AP.Compute.Equalize.Data

12 Property

Syntax **AP.Compute.Equalize.Data(*source%, column%*)**

Data Type Boolean

Parameters

Name	Description
source%	Number of the Sweep Data (1-6) of the data in memory.
column%	Number of the Data Column (0-7) of the data specified by the AP.Compute.Equalize.FileName command.

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

See Also AP.Compute.Equalize.Apply, AP.Compute.Equalize.FileName

Example See example for AP.Compute.Equalize.Apply.

AP.Compute.Equalize.FileName

12 Property

Syntax **AP.Compute.Equalize.FileName(*filename$*)**

Parameters

Name	Description
filename$	Any valid DOS filename and extension. Enter "SweepData" for the file name to select data in memory.

Result Boolean

Description

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

See Also `AP.Compute.Equalize.Apply`

Example See example for `AP.Compute.Equalize.Apply`.

AP.Compute.Equalize.PostSweep ❶❷ Property

Syntax **`AP.Compute.Equalize.PostSweep`**

Parameters None

Data Type Boolean

`True`	Enable computation to be applyed 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.

APWIN 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 `AP.Compute.Equalize.Apply`

Example See example for `AP.Compute.Equalize.Apply`.

AP.Compute.Invert.Apply ❶❷ Method

Syntax **`AP.Compute.Invert.Apply`**

Parameters None

Result Boolean

True	Computation performed.
False	Computation NOT performed.

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

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

❷ Example

```
Sub Main
   AP.File.OpenTest "INVERT1.AT2"      'Open test.
   AP.Compute.Clear.All       'Clear Compute functions.
   AP.Compute.Invert.PostSweep = False 'Post Sweep Off.
   AP.Compute.Invert.Data(1) = True 'Data to be inverted.
   AP.Compute.Invert.Horizontal("Hz") = 5000'Horizontal _
      Value.
   AP.Sweep.Start
   AP.Compute.Invert.Apply
End Sub
```

AP.Compute.Invert.Data

❶❷ Property

Syntax **`AP.Compute.Invert.Data(`***`column%`***`)`**

Data Type Boolean

Parameters

Name	Description
column%	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 columns, several Invert computations can be performed in one operation.

See Also `AP.Compute.Invert.Apply`

Example See example for `AP.Compute.Invert.Apply`.

AP.Compute.Invert.Horizontal ❶❷ Property

Syntax **`AP.Compute.Invert.Horizontal`**

Data Type Double

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

See Also `AP.Compute.Invert.Apply`

Example See example for `AP.Compute.Invert.Apply`.

AP.Compute.Invert.PostSweep ❶❷ Property

Syntax **`AP.Compute.Invert.PostSweep`**

Parameters None

Data Type Boolean

`True`	Enable computation to be applyed after sweep.
`False`	Disable computation after sweep.

Description 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.

APWIN 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 `AP.Compute.Invert.Apply`

Example See example for `AP.Compute.Invert.Apply`.

AP.Compute.Linearity.Apply ❶❷ Method

Syntax **`AP.Compute.Linearity.Apply`**

Parameters None

Result Boolean

`True`	Computation performed.
`False`	Computation NOT performed.

Description This command applies the Linearity computation to the selected data (1-6). The difference

See Also `AP.Compute.Clear.All, AP.Compute.Linearity.Data, AP.Compute.Linearity.PostSweep, AP.Compute.Linearity.Start, AP.Compute.Linearity.Stop,`

❷ **Example**

```
Sub Main
   AP.File.OpenTest "LINEAR1.AT2"      'Open test.
   AP.Compute.Clear.All
   AP.Compute.Linearity.PostSweep = False'Disables _
      Apply after sweep.
   AP.Compute.Linearity.Data(1) = True 'Use column 1 _
      for data 1.
   AP.Compute.Linearity.Start("Vrms") = .5 'Start at _
      500mV.
   AP.Compute.Linearity.Stop("Vrms") = 2  'Stop at 2V.
   AP.Sweep.Start
   AP.Compute.Linearity.Apply    'Start computation.
   AP.Sweep.Data1.Bottom("V") = -.02
   AP.Sweep.Data1.Top("V") = .02
End Sub
```

AP.Compute.Linearity.Data ❶❷ Property

Syntax `AP.Compute.Linearity.Data(`*column%*`)`

Data Type Boolean

Parameters

Name	Description
column%	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 columns, several Linearity computations can be performed in one operation.

See Also `AP.Compute.Linearity.Apply`

Example See example for `AP.Compute.Linearity.Apply`.

AP.Compute.Linearity.PostSweep ❶❷ Property

Syntax `AP.Compute.Linearity.PostSweep`

Parameters None

Data Type Boolean

True	Enable computation to be applyed 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.

APWIN 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 `AP.Compute.Linearity.Apply`

Example See example for `AP.Compute.Linearity.Apply`.

AP.Compute.Linearity.Start

1 2 Property

Syntax **`AP.Compute.Linearity.Start`**

Data Type Double

Parameters

Name	Description
unit$	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 `AP.Compute.Linearity.Stop`, `AP.Compute.Linearity.Apply`

Example See example for `AP.Compute.Linearity.Apply`.

AP.Compute.Linearity.Stop

1 2 Property

Syntax **`AP.Compute.Linearity.Stop`**

Data Type Double

Parameters

Name	Description
unit$	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 `AP.Compute.Linearity.Start`, `AP.Compute.Linearity.Apply`

Example See example for `AP.Compute.Linearity.Apply`.

AP.Compute.Max.Apply ❶❷ Method

Syntax **AP.Compute.Max.Apply**

Parameters None

Result Boolean

True	Computation performed.
False	Computation NOT performed.

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

See Also AP.Compute.Clear.All, AP.Compute.Max.Data, AP.Compute.Max.PostSweep, AP.Compute.Max.Start, AP.Compute.Max.Stop,

❷ Example

```
Sub Main
   AP.File.OpenTest "MAX1.AT2"      'Open test.
   AP.Compute.Clear.All
   AP.Compute.Max.PostSweep = False 'Disables Apply _
      after Sweep.
   AP.Compute.Max.Data(1) = True 'Use column 1 for _
      data 1.
   AP.Compute.Max.Start("Hz") = 2000
   AP.Compute.Max.Stop("Hz") = 200
   AP.Sweep.Start
   AP.Compute.Max.Apply
End Sub
```

AP.Compute.Max.Data ❶❷ Property

Syntax **AP.Compute.Max.Data(***column%***)**

Data Type Boolean

Parameters

Name	Description
column%	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 columns, several Maximum computations can be performed in one operation.

See Also `AP.Compute.Max.Apply`

Example See example for `AP.Compute.Max.Apply`.

AP.Compute.Max.PostSweep

❶❷ Property

Syntax **`AP.Compute.Max.PostSweep`**

Parameters None

Data Type Boolean

`True`	Enable computation to be applyed 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.

APWIN 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 `AP.Compute.Max.Apply`

Example See example for `AP.Compute.Max.Apply`.

AP.Compute.Max.Start

❶❷ Property

Syntax **`AP.Compute.Max.Start`**

Data Type Double

Parameters	Name	Description
	unit$	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 `AP.Compute.Max.Stop, AP.Compute.Max.Apply`

Example See example for `AP.Compute.Max.Apply`.

AP.Compute.Max.Stop

❶❷ Property

Syntax **`AP.Compute.Max.Stop`**

Data Type Double

Parameters	Name	Description
	unit$	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 Maximum computation will be performed.

See Also `AP.Compute.Max.Start, AP.Compute.Max.Apply`

Example See example for `AP.Compute.Max.Apply`.

AP.Compute.Min.Apply

❶❷ Method

Syntax **`AP.Compute.Min.Apply`**

Parameters None

Result Boolean

	True	Computation performed.

False	Computation NOT performed.

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

See Also AP.Compute.Clear.All, AP.Compute.Min.Data, AP.Compute.Min.PostSweep, AP.Compute.Min.Start, AP.Compute.Min.Stop,

❷ Example

```
Sub Main
   AP.File.OpenTest "MIN1.AT2"          'Open test.
   AP.Compute.Clear.All
   AP.Compute.Min.PostSweep = False 'Disables apply _
      after sweep.
   AP.Compute.Min.Data(1) = True 'Use column 1 for _
      data 1.
   AP.Compute.Min.Start("Hz") = 10000
   AP.Compute.Min.Stop("Hz") = 200
   AP.Sweep.Start
   AP.Compute.Min.Apply
End Sub
```

AP.Compute.Min.Data

❶❷ Property

Syntax **AP.Compute.Min.Data(***column%***)**

Data Type Boolean

Parameters

Name	Description
column%	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 columns, several Minmum computations can be performed in one operation.

See Also `AP.Compute.Min.Apply`

Example See example for `AP.Compute.Min.Apply`.

AP.Compute.Min.PostSweep ❶❷ Property

Syntax **`AP.Compute.Min.PostSweep`**

Parameters None

Data Type Boolean

`True`	Enable computation to be applyed 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.

APWIN 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 `AP.Compute.Min.Apply`

Example See example for `AP.Compute.Min.Apply`.

AP.Compute.Min.Start ❶❷ Property

Syntax **`AP.Compute.Min.Start(`***`unit$`***`)`**

Data Type Double

Parameters

Name	Description
`unit$`	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 `AP.Compute.Min.Stop, AP.Compute.Min.Apply`

Example See example for `AP.Compute.Min.Apply`.

AP.Compute.Min.Stop ❶❷ Property

Syntax **`AP.Compute.Min.Stop`**

Data Type Double

Parameters

Name	Description
`unit$`	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 Minimum computation will be performed.

See Also `AP.Compute.Min.Start, AP.Compute.Min.Apply`

Example See example for `AP.Compute.Min.Apply`.

AP.Compute.Normalize.Apply ❶❷ Method

Syntax **`AP.Compute.Normalize.Apply`**

Parameters None

Result Boolean

`True`	Computation performed.
`False`	Computation NOT performed.

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

See Also
```
AP.Compute.Clear.All, AP.Compute.Normalize.Data,
AP.Compute.Normalize.Horizontal,
AP.Compute.Normalize.PostSweep,
AP.Compute.Normalize.Target
```

❷ **Example**
```
Sub Main
```

```
   AP.File.OpenTest "NORMAL1.AT2"'opens test to be run.
   AP.Compute.Clear.All
   AP.Compute.Normalize.PostSweep = False 'Disables _
      apply after sweep.
   AP.Compute.Normalize.Data(1) = True
   AP.Compute.Normalize.Horizontal("Hz") = 1000 'Sets _
      1kHz point to be normalized.
   AP.Compute.Normalize.Target("dBV") = 0.0 'Normalize _
      1Khz point to 0.0dBV.
   AP.Sweep.Start
   AP.Compute.Normalize.Apply
End Sub
```

AP.Compute.Normalize.Data ❶❷ Property

Syntax **AP.Compute.Normalize.Data(*column%*)**

Data Type Boolean

Parameters

Name	Description
column%	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 columns, several Normalize computations can be performed in one operation.

See Also AP.Compute.Normalize.Apply

Example See example for AP.Compute.Normalize.Apply.

AP.Compute.Normalize.Horizontal ❶❷ Property

Syntax `AP.Compute.Normalize.Horizontal(`*unit$*`)`

Data Type Double

Parameters

Name	Description
unit$	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 `AP.Compute.Normalize.Apply`, `AP.Compute.Normalize.Target`

Example See example for `AP.Compute.Normalize.Apply`.

AP.Compute.Normalize.PostSweep ❶❷ Property

Syntax `AP.Compute.Normalize.PostSweep`

Data Type

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.

APWIN 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 `AP.Compute.Normalize.Apply`

Example See example for `AP.Compute.Normalize.Apply`.

AP.Compute.Normalize.Target ❶❷ Property

Syntax `AP.Compute.Normalize.Target(unit$)`

Data Type Double

Parameters

Name	Description
unit$	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 `AP.Compute.Normalize.Apply`, `AP.Compute.Normalize.Horizontal`

Example See example for `AP.Compute.Normalize.Apply`.

AP.Compute.Sigma.Apply ❶❷ Method

Syntax `AP.Compute.Sigma.Apply`

Parameters None

Result Boolean

True	Computation performed.
False	Computation NOT performed.

Description This command applies the 2-Sigma computation to the selected data (1-6).

See Also `AP.Compute.Clear.All, AP.Compute.Sigma.Data, AP.Compute.Sigma.PostSweep, AP.Compute.Sigma.Start, AP.Compute.Sigma.Stop,`

❷ **Example**

```
Sub Main
   Dim status As Boolean
   status = AP.Log.Enable     'Get logging condition.
   AP.Log.Enable = False      'Turn logging off.
   AP.File.OpenTest "SIGNAL.AT2" 'Open test.
   AP.Compute.Clear.All
```

```
   AP.Compute.Sigma.PostSweep = False 'Disables apply _
      after sweep.
   AP.Compute.Sigma.Data(1) = True 'Set data 1 for _
      Compute Sigma.
   AP.Compute.Sigma.Start("sec") = 6
   AP.Compute.Sigma.Stop("sec") = 12
   AP.Sweep.Start
   AP.Compute.Sigma.Apply
   AP.Log.Enable = status     'Return to initial _
logging condition.
End Sub
```

AP.Compute.Sigma.Data

❶❷ Property

Syntax **AP.Compute.Sigma.Data(***column%***)**

Data Type Boolean

Parameters

Name	Description
column%	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 2-Sigma computation is to be performed on. By using this command several times to select multiple columns, several 2-Sigma computations can be performed in one operation.

See Also AP.Compute.Sigma.Apply

Example See example for AP.Compute.Sigma.Apply.

AP.Compute.Sigma.PostSweep ❶❷ Property

Syntax `AP.Compute.Sigma.PostSweep`

Parameters None

Data Type Boolean

True	Enable computation to be applyed after sweep.
False	Disable computation after sweep.

Description This command instructs the test to perform the 2-Sigma computation after a sweep is complete and sets the state of the Apply After Sweep field on the Compute 2-Sigma panel.

APWIN 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 `AP.Compute.Sigma.Apply`

Example See example for `AP.Compute.Sigma.Apply`.

AP.Compute.Sigma.Start ❶❷ Property

Syntax `AP.Compute.Sigma.Start`

Data Type Double

Parameters

Name	Description
unit$	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 2-Sigma computation will be performed.

See Also `AP.Compute.Sigma.Stop, AP.Compute.Sigma.Apply`

Example See example for `AP.Compute.Sigma.Apply`.

AP.Compute.Sigma.Stop ❶❷ Property

Syntax `AP.Compute.Sigma.Stop`

Data Type Double

Parameters

Name	Description
unit$	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 2-Sigma computation will be performed.

See Also `AP.Compute.Sigma.Start, AP.Compute.Sigma.Apply`

Example See example for `AP.Compute.Sigma.Apply`.

AP.Compute.Smooth.Apply ❶❷ Method

Syntax `AP.Compute.Smooth.Apply`

Parameters None

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 `AP.Compute.Clear.All, AP.Compute.Smooth.Data, AP.Compute.Smooth.Passes, AP.Compute.Smooth.PostSweep`

❷ **Example**

```
Sub Main
   AP.File.OpenTest "SMOOTH1.AT2"      'Open test.
   AP.Compute.Clear.All
   AP.Compute.Smooth.PostSweep = False 'Disable Apply _
      after Sweep.
   AP.Compute.Smooth.Auto = False'Disable Auto Smoothing.
   AP.Compute.Smooth.Data(1) = True
```

```
    AP.Compute.Smooth.Passes = 1 'Set Smooth Passes to 1.
    AP.Sweep.Start
    AP.Compute.Smooth.Apply
End Sub
```

AP.Compute.Smooth.Auto ❶❷ Property

Syntax **AP.Compute.Smooth.Auto**

Parameters None

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 AP.Compute.Smooth.Apply

Example See example for AP.Compute.Smooth.Apply.

AP.Compute.Smooth.Data ❶❷ Property

Syntax **AP.Compute.Smooth.Data(***column%***)**

Data Type Boolean

Parameters

Name	Description
column%	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 columns, several Smooth computations can be performed in one operation.

See Also `AP.Compute.Smooth.Apply`

Example See example for `AP.Compute.Smooth.Apply`.

AP.Compute.Smooth.Passes

❶❷ Property

Syntax **`AP.Compute.Smooth.Passes`**

Parameters None

Data Type Long

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

See Also `AP.Compute.Smooth.Apply, AP.Compute.Smooth.Data`

Example See example for `AP.Compute.Smooth.Apply`.

AP.Compute.Smooth.PostSweep

❶❷ Property

Syntax **`AP.Compute.Smooth.PostSweep`**

Parameters None

Data Type Boolean

`True`	Enable computation to be applyed 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.

APWIN 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 `AP.Compute.Smooth.Apply`

Example See example for `AP.Compute.Smooth.Apply`.

User Notes

User Notes

User Notes

User Notes

User Notes

Data

AP.Data.AddRowToEnd ❶❷ Method

Syntax **AP.Data.AddRowToEnd(*id%*)**

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to AP.Data.Id 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
   AP.Application.NewTest
   AP.Sweep.Data2.Id = 5905 'Add Phase Data to Sweep

   StartValue = 20
   StopValue = 20000
   Frequencies = 31

   Counter = StartValue
   Do
      intAddRowToEnd = AP.Data.AddRowToEnd(0) 'Add row
      AP.Data.Value(0,0,intAddRowToEnd,"Hz") = Counter
      AP.Data.Value(0,1,intAddRowToEnd,"V") = 1.0
      AP.Data.Value(0,2,intAddRowToEnd,"deg") = 0.0
      Counter = Counter * 1.25893         'Log spacing
   Loop Until Counter > StopValue

   intAddRowToEnd = AP.Data.AddRowToEnd(0)
   AP.Data.Value(0,0,intAddRowToEnd,"Hz") = StopValue
   AP.Data.Value(0,1,intAddRowToEnd,"V") = 1.0
   AP.Data.Value(0,2,intAddRowToEnd,"deg") = 0.0
   AP.Data.UpdateDisplay(0)
End Sub
```

AP.Data.ColLimitError ❶❷ Method

Syntax **AP.Data.ColLimitError(***id%, column%***)**

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to AP.Data.Id command for additional information.
column%	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 AP.Data.LimitError

❶ **Example**

```
Sub Main
   AP.File.OpenTest “S1-FREQ.AT1"
   AP.Sweep.Start
   Errors = AP.Data.ColLimitError(0,1)
   If Errors > 0 Then
      ErrorsUpper = AP.Data.ColUpperLimitError(0,1)
      ErrorsLower = AP.Data.ColLowerLimitError(0,1)
      String1$ = “This test Failed. ” & Str(Errors) _
         & “ Errors.”
      String2$ = Str(ErrorsUpper)&“ Upper Limit Errors.”
      String3$ = Str(ErrorsLower)&" Lower Limit Errors.“
      AP.Prompt.Text = String1$ & Chr(13) & String2$ & _
         Chr(13) & String3$
      AP.Prompt.FontSize = 18
      AP.Prompt.Position -1,-1,425,175
      AP.Prompt.ShowWithContinue
      Stop
   ElseIf Errors = 0 Then
      AP.Prompt.Text = “This test Passed.”
      AP.Prompt.FontSize = 18
      AP.Prompt.Position -1,-1,290,100
      AP.Prompt.ShowWithContinue
      Stop
```

```
   End If
End Sub
```


AP.Data.ColLowerLimitError

❶❷ Method

Syntax **AP.Data.ColLowerLimitError(*id%, column%*)**

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to AP.Data.Id command for additional information.
column%	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 AP.Data.ColLimitError.

AP.Data.ColName

❶❷ Method

Syntax **AP.Data.ColName(*id%, column%*)**

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to AP.Data.Id command for additional information.
column%	Number of the Data Column (0-7).

Result String

Description This command returns the string as shown on the sweep panel for the selected data. The string defines the meter that is returning measurements.

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
```

```
    AP.Sweep.Start
    ColumnName$ = AP.Data.ColName(0,1)
    Debug.Print "String definition For Data 1: = " & _
       ColumnName$
End Sub
```

Comment This macro puts up a prompt displaying the contents of the Data 1 control on the Sweep Panel.

Example Output

```
String definition For Data 1: = S1.Anlr.Level A
```

AP.Data.ColNumOf

❶❷ Method

Syntax **AP.Data.ColNumOf(*id%*)**

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to AP.Data.Id command for additional information.

Result Integer

Description This command returns the number of columns of data.

Example See example for AP.Data.ColSize.

AP.Data.ColSize

❶❷ Method

Syntax **AP.Data.ColSize(*id%, column%*)**

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to AP.Data.Id command for additional information.
column%	Number of the Data Column (0-7).

Result Long

Description This command returns the number of rows in the specified column.

Example

```
Sub Main
```

```
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Sweep.Start
   NumColumns = AP.Data.ColNumOf(0)
   For Column = 1 To (NumColumns - 1) Step 1
      Size = AP.Data.ColSize(0, Column)
      Debug.Print "Number of measurements for Column _
         ";Column;" = "; Size
   Next Column
End Sub
```

Example Output

```
Number of measurements for Column 1 = 31
```

AP.Data.ColUnit ❶❷ Property

Syntax **AP.Data.ColUnit(*id%, column%*)**

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to AP.Data.Id command for additional information.
column%	Number of the Data Column (0-7).

Result String

Description This command returns the unit string for the data in the selected column. APWIN computes all other relevant units for display.

Example

```
Sub Main
   AP.Application.NewTest     'Reset panels
   AP.Gen.Output = 1          'Turn Generator output ON
   AP.Anlr.ChAInput = 2       'Set Analyzer input to _
      GENMON
   AP.Sweep.SinglePoint = 1   'Set sweep for single _
      measurement
   AP.Sweep.Start             'Run sweep
   Unit = AP.Data.ColUnit(0, 1)
   Debug.Print "The Unit for Data 1 is " & Unit
End Sub
```

AP.Data.ColUpperLimitError ❶❷ Method

Syntax `AP.Data.ColUpperLimitError(id%, column%)`

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to AP.Data.Id command for additional information.
column%	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 AP.Data.ColLimitError.

AP.Data.DeleteRow ❶❷ Method

Syntax `AP.Data.DeleteRow(id%, row%)`

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to AP.Data.Id command for additional information.
row%	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

   DataSize = AP.Data.ColSize(0,0)
   ReDim FreqData(DataSize)
   FreqData = AP.Data.XferToArray(0,0,"Hz")
```

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

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

AP.Data.Id

Method

Syntax **AP.Data.Id(***filename$***)**

Data Type String

Parameters

Name	Description

filename$	Any valid DOS path and file name. The file must be an APWIN limit, sweep, or data (.adl, .ads, .ada) file attached to the current test.

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 `AP.Data.ColLimitError, AP.Data.ColLowerLimitError, AP.Data.ColName, AP.Data.ColNumOf, AP.Data.ColSize, AP.Data.ColUpperLimitError, AP.Data.LimitError, AP.Data.LowerLimitError, AP.Data.OptimizeDisplay, AP.Data.UpdateDisplay, AP.Data.UpperLimitError, AP.Data.Value, AP.Data.XferToArray`

❷ **Example**

```
Sub Main
   Dim Limitarray As Double
   Dim Tablearray As Double
   AP.File.OpenTest “ID.AT2"                'Open test
   LimitId = AP.Data.Id(“C:\APWIN\APBASIC\LIMIT.ADL”)
   TableId = AP.Data.Id(“C:\APWIN\APBASIC\TABLE.ADS”)
   LimitArray = AP.Data.XferToArray(LimitId, 1, “V”)
   TableArray = AP.Data.XferToArray(TableId, 1, “V”)
   Debug.Print “Limit ID # = ”;LimitId
   Debug.Print “Table ID # = ”;TableId
End Sub
```

Example Output

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

AP.Data.InsertRowAfter ❶❷ Method

Syntax **`AP.Data.InsertRowAfter(`***`id%, row%`***`)`**

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to `AP.Data.Id` command for additional information.

	row%	Number of row to insert after.

Result Boolean

True	Row inserted.
False	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 `AP.Data.InsertRowBefore`.

AP.Data.InsertRowBefore

❶❷ Method

Syntax **`AP.Data.InsertRowBefore(`***id%, row%***`)`**

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to `AP.Data.Id` command for additional information.
row%	Number of row to insert before.

Result Boolean

True	Row deleted.
False	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 `AP.Data.InsertRowAfter`.

AP.Data.LimitError

❶❷ Method

Syntax **`AP.Data.LimitError(`***id%***`)`**

Parameters

Name	Description

id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to `AP.Data.Id` command for additional information.

Result Integer

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

❶ Example

```
Sub Main
   AP.File.OpenTest "S1-FREQ.AT1"
   AP.Sweep.Start
   Errors = AP.Data.LimitError(0)
   If Errors > 0 Then
      ErrorsUpper = AP.Data.ColUpperLimitError(0,1)
      ErrorsLower = AP.Data.ColLowerLimitError(0,1)
      String1$ = "This test Failed. " & Str(Errors) & _
         " Errors. "
      String2$ = Str(ErrorsUpper)&" Upper Limit Errors."
      String3$ = Str(ErrorsLower)&" Lower Limit Errors."
      AP.Prompt.Text = String1$ & Chr(13) & String2$ & _
         Chr(13) & String3$
      AP.Prompt.FontSize = 18
      AP.Prompt.Position -1,-1,425,175
      AP.Prompt.ShowWithContinue
      Stop
   ElseIf Errors = 0 Then
      AP.Prompt.Text = "This test Passed."
      AP.Prompt.FontSize = 18
      AP.Prompt.Position -1,-1,290,100
      AP.Prompt.ShowWithContinue
      Stop
   End If
End Sub
```

AP.Data.LowerLimitError ❶❷ Method

Syntax `AP.Data.LowerLimitError(`*id%*`)`

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to `AP.Data.Id` 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.

❶ Example

```
Sub Main
   AP.File.OpenTest "S1-FREQ.AT1"
   AP.Sweep.Start
   Flag = AP.Data.LowerLimitError(0)
   If Flag > 0 Then
      AP.Prompt.Text = "This test Failed."
      AP.Prompt.FontSize = 18
      AP.Prompt.Position -1,-1,290,100
      AP.Prompt.ShowWithContinue
      Stop
   ElseIf Flag = 0 Then
      AP.Prompt.Text = "This test Passed."
      AP.Prompt.FontSize = 18
      AP.Prompt.Position -1,-1,290,100
      AP.Prompt.ShowWithContinue
      Stop
   End If
End Sub
```

AP.Data.OptimizeDisplay ❶❷ Method

Obsolete Obsolete command not recomended for new design.

Syntax `AP.Data.OptimizeDisplay(id%)`

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to `AP.Data.Id` command for additional information.

Result Void

Description This command optimizes the graph to display all data.

AP.Data.UpdateDisplay ❶❷ Method

Syntax `AP.Data.UpdateDisplay(id%)`

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data only.

Result Void

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

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.FuncFilterLP = 1
   AP.Sweep.Data1.Id = 5906
   AP.Sweep.Data2.Id = 5906
   AP.Sweep.Source1.Start("Hz") = 100000
   AP.Sweep.Source1.Steps = 4
   AP.Sweep.Start
   Size = AP.Data.ColSize (0, 1)
   For Reading = 0 To (Size - 1) Step 1   'Read readings.
```

```
            Debug.Print "Acquired Reading";Reading;" = " _
               ;Format(AP.Data.Value(0, 1, Reading), _
               "#.0000");" V" 'Return reading.
         Next Reading
         For Reading = 0 To (Size - 1) Step 1
            Measurement = AP.Data.Value(0, 1, Reading)
            AP.Data.Value(0, 2, Reading) = (Measurement * _
               1.20) 'Increase level by 20%.
         Next Reading
         For Reading = 0 To (Size - 1) Step 1
            Debug.Print "Limit";Reading;" =" _
               ;Format(AP.Data.Value(0, 2, Reading), _
               "#.0000");" V" 'Return reading.
         Next Reading
         AP.Data.UpdateDisplay(0)
         AP.File.SaveDataAs "UPPER.ADL"
      End Sub
```

Example 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
Limit 0 = .0262 V
Limit 1 = 1.1848 V
Limit 2 = 1.1925 V
Limit 3 = 1.1920 V
Limit 4 = 1.1940 V
```

AP.Data.UpperLimitError

❶❷ Method

Syntax `AP.Data.UpperLimitError(`*id%*`)`

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to `AP.Data.Id` 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.

❶ **Example**

```
Sub Main
   AP.File.OpenTest "S1-FREQ.AT1"
   AP.Sweep.Start
   Flag = AP.Data.UpperLimitError(0)
   If Flag > 0 Then
      AP.Prompt.Text = "This test Failed."
      AP.Prompt.FontSize = 18
      AP.Prompt.Position -1,-1,290,100
      AP.Prompt.ShowWithContinue
      Stop
   ElseIf Flag = 0 Then
      AP.Prompt.Text = "This test Passed."
      AP.Prompt.FontSize = 18
      AP.Prompt.Position -1,-1,290,100
      AP.Prompt.ShowWithContinue
      Stop
   End If
End Sub
```

AP.Data.Value ❶❷ Property

Syntax **AP.Data.Value(***id%, column%, row&, unit$***)**

Data Type Double

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to AP.Data.Id command for additional information.
column%	Number of the Data Column (0-7).
row&	This value defines which row a measurement is returned from. A column may have any number of rows. Use the AP.Data.ColSize command to determine the number

	of rows in a column. Note that the row numbering begins with zero.
unit$	Refer to the setting or reading defined by the *column parameter to determine t e appropriate unit selections.*

Description This command returns the specified reading from sweep data.

See Also `AP.Data.ColSize, AP.Data.Id`

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = 1    'Turn Generator output ON
   AP.Anlr.ChAInput = 2 'Set Analyzer input to GENMON
   AP.Sweep.SinglePoint = 1 'Set sweep for single _
      measurement
   AP.Sweep.Start       'Run sweep
   Reading1 = AP.Data.Value(0, 1, 0, "V") 'Get Reading
   Debug.Print "Reading = ";Format(Reading1, _
      "#.0000"); "V"
End Sub
```

Example Output

```
Reading = .9850 V
```

AP.Data.XferToArray ❶❷ Property

Syntax **`AP.Data.XferToArray(`***`id%,`*** `column%,` ***`unit$`*****`)`**

Data Type Variant

Parameters

Name	Description
id%	Data identification number. Use an Id# of zero (0) to access sweep data. Refer to `AP.Data.Id` command for additional information.
column%	Number of the Data Column (0-7).
unit$	Refer to the setting or reading defined by the *column% parameter to determine t e appropriate unit selections.*

Description This command transfers the contents of a column (Sweep Source 1-2 or Data 1-6) to an array.

See Also `AP.Data.Id`

Example

```
Sub Main
   Dim A As Variant
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.FuncFilterLP = 1
   AP.Sweep.Data1.Id = 5906
   AP.Sweep.Source1.Start("Hz") = 100000
   AP.Sweep.Source1.Steps = 4
   AP.Application.NewData  'Clear data loaded with test.
   AP.Sweep.Start
   Size = AP.Data.ColSize (0, 1)
                           'Transfer data to array.
   A = AP.Data.XferToArray(0, 1, "V")
   For Reading = 0 To (Size - 1) Step 1
      Debug.Print "Reading ";Reading;" = "; _
         Format(A(Reading), "#.0000");" V"
   Next Reading
End Sub
```

Example Output

```
Reading  0 = .0223 V
Reading  1 = .9889 V
Reading  2 = .9953 V
Reading  3 = .9945 V
Reading  4 = .9931 V
```

Comment

The values in the example output are taken from the array and then displayed.

User Notes

User Notes

User Notes

User Notes

DCX-127

AP.DCX.Ch1DcLevel ❶❷ Property

Syntax `AP.DCX.Ch1DcLevel`

Data Type Double -10.5 to 10.5 Volts

Description This command sets the voltage at the DCX's channel 1 DC output.

❶ **Example**

```
Sub Main
   AP.File.OpenTest "DCX1.AT1"      'Opens test.
   AP.DCX.Ch1DcOutput = 1
   AP.DCX.Ch1DcLevel("Vdc") = 1.5
End Sub
```

Comment This macro turns on the DCX's Channel 1 DC output and sets it to 1.5 volts.

AP.DCX.Ch1DcOutput ❶❷ Property

Syntax `AP.DCX.Ch1DcOutput`

Data Type Boolean

True	Connects the output to the front panel.
False	Disconnects the output from the front panel.

Description This command sets DC Volts output 1 to On or Off.

Example See example for `AP.DCX.Ch1DcLevel`.

AP.DCX.Ch2DcLevel ❶❷ Property

Syntax `AP.DCX.Ch2DcLevel`

Data Type Double -10.5 to 10.5 Volts

Description This command sets the voltage at the DCX's channel 2 DC output.

❶ Example

```
Sub Main
   AP.File.OpenTest "DCX1.AT1"      'Open test.
   AP.DCX.Ch2DcOutput = 1
   AP.DCX.Ch2DcLevel("Vdc") = 1.5
End Sub
```

Comment This macro turns on the DCX's Channel 2 DC output and sets it to 1.5 volts.

AP.DCX.Ch2DcOutput ❶❷ Property

Syntax `AP.DCX.Ch2DcOutput`

Data Type Boolean

True	Connects the output to the front panel.
False	Disconnects the output from the front panel.

Description This command sets DC Volts output 2 to On or Off.

Example See example for `AP.DCX.Ch2DcLevel`.

AP.DCX.DigInFormat ❶❷ Property

Syntax `AP.DCX.DigInFormat`

Data Type Integer

0	2's Complement
1	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 3FFFFF 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
   AP.File.OpenTest "DCX1.AT1"    'Open test.
   AP.DCX.DigOutFormat = 1        'Sets the digital _
      output format to BCD.
   AP.DCX.DigInRdgRate = 1        'Selects input strobe _
      rate of 4/sec.
   AP.DCX.DigOut("dec") = 100     'Sets the digital _
      output to 100 dec.
   AP.DCX.DigInFormat = 1         'Sets format of _
      digital input to BCD.
   Reading1 = AP.DCX.DigInRdg("dec")   'Returns a _
      settled reading 100 in dec.
   AP.DCX.DigOut("h(x)") = 100    'Sets the digital _
      output to 100 dec scaled.
   AP.DCX.DigOutScale = 2         'Scales the digital _
      output by 2.
   AP.DCX.DigInSettling(.20, .1, "Dec", 4, .05, 0)
   AP.DCX.DigInTrig               'Trigger a new reading.
   Do
      Ready = AP.DCX.DigInReady
   Loop Until Ready > 0           'Wait until new _
         reading is ready.
   Reading2 = AP.DCX.DigInRdg("dec")'Returns a settled _
      reading 200 in dec.
   AP.DCX.DigInScale = .5         'Scales the digital _
      input by .5.
   Reading3 = AP.DCX.DigInRdg("g(x)")  'Returns  _
      a settled reading 100 in dec.


   NewLine$ = Chr(13)
   a$= "Reading1 "+Left(Str$(Reading1),6)+"dec"
   b$= "Reading2 "+Left(Str$(Reading2),6)+"dec"
   c$= "Reading3 "+Left(Str$(Reading3),6)+"dec"
   AP.Prompt.Text = a$ + NewLine$ + b$ + NewLine$ + c$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.DCX.DigInRdg

12 Property

Syntax `AP.DCX.DigInRdg(unit$)`

Data Type Variant

Parameters

Part	Description
unit$	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 `AP.DCX.DigInReady, AP.DCX.DigInSettling, AP.DCX.DigInTrig`

Example See example for `AP.DCX.DigInFormat`.

AP.DCX.DigInRdgRate

12 Property

Syntax `AP.DCX.DigInRdgRate`

Data Type Integer

0	External Strobe (Default).
1	4 readings per second.
2	8 readings per second.
3	16 readings per second.
4	32 readings per second.

Description 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.

Example See example for `AP.DCX.DigInFormat`.

AP.DCX.DigInReady

12 Property

Syntax `AP.DCX.DigInReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description 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 `AP.DCX.DigInRdg` or `AP.DCX.DigInTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.DCX.DigInRdg` command will be guaranteed to return quickly.

See Also `AP.DCX.DigInRdg, AP.DCX.DigInSettling, AP.DCX.DigInTrig`

Example See example for `AP.DCX.DigInFormat`.

AP.DCX.DigInScale ❶❷ Property

Syntax **`AP.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, APWIN software computes the displayed value from the relationship

display = measurement * 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 `AP.DCX.DigInFormat`.

AP.DCX.DigInSettling ❶❷ Method

Syntax **AP.Anlr.PhaseSettling(***tolerance#, floor#, floorunit$, points%, delay#, algorithm%***)**

Description This command sets the settling parameters for the AP.Dcx.DigInRdg command.

See Appendix C for Settling Algorithm and command part descriptions.

See Also AP.DCX.DigInFormat, AP.DCX.DigInRdg, AP.DCX.DigInReady, AP.DCX.DigInTrig

Example See example for AP.DCX.DigInFormat.

AP.DCX.DigInTrig ❶❷ Method

Syntax **AP.DCX.DigInTrig**

Description Causes a restart of the reading cycle and zeros the ready count for the AP.DCX.DigInRdg command. The reading in progress is aborted.

See Also AP.DCX.DigInRdg, AP.DCX.DigInReady, AP.DCX.DigInSettling

Example See example for AP.DCX.DigInFormat.

AP.DCX.DigOut ❶❷ Property

Syntax **AP.DCX.DigOut(***units$***)**

Data Type Double

Parameters

Name	Description
units$	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 AP.DCX.DigOutFormat command.

Example See example for `AP.DCX.DigInFormat`.

AP.DCX.DigOutFormat ❶❷ Property

Syntax **`AP.DCX.DigOutFormat`**

Data Type Integer

0	2's Complement
1	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 3FFFFF 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 `AP.DCX.DigInFormat`.

AP.DCX.DigOutScale ❶❷ Property

Syntax **`AP.DCX.DigOutScale`**

Data Type Double

Description When h(x) units are selected at the Digital Output control field, APWIN software computes the actual transmitted value from the relationship

output value = entry value* 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 `AP.DCX.DigInFormat`.

AP.DCX.DmmMode

❶❷ Property

Syntax `AP.DCX.DmmMode`

Data Type Integer

0	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 by System One or System Two.
1	DC Volts
2	Ohms

Description This command sets the DMM measurement mode.

❶ **Example**

```
Sub Main
   AP.File.OpenTest “DCX1.AT1" ‘Open test.
   AP.DCX.DmmRange = 2.0       ‘set DMM input to _
      2 Volt range.
   AP.DCX.DmmMode = 1       ‘set DMM mode to volts.
   AP.DCX.DmmRdgRate = 0    ‘set DMM reading rate to 6 _
      readings per second.
   AP.DCX.DmmSettling(1, .20, “Vdc”, 3, .03, 0) ‘Set _
      settling parameters.

   AP.DCX.DmmTrig           ‘Trigger a new reading.
   Do
      Ready =  AP.DCX.DmmReady
   Loop Until Ready > 0 ‘Loop until new reading is ready
   Reading1 =  AP.DCX.DmmRdg(“Vdc”) ‘Returns a settled _
      reading.

   NewLine$ = Chr(13)
   a$= “DMM Reading ”+Left(Str$(Reading1),6)+"Vdc"
   AP.Prompt.Text = a$ + NewLine$
   AP.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.

AP.DCX.DmmOffset ❶❷ Property

Syntax `AP.DCX.DmmOffset`

Data Type Double

Description When f(V) (function of Volts) or f(O) (function of Ohms) units are selected for the DMM, APWIN software computes the value to display from the formula

```
display = (measurement + Offset) * 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 `AP.DCX.DmmScale`

Example See example for `AP.DCX.DmmMode`.

AP.DCX.DmmRange ❶❷ Property

Syntax `AP.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 `AP.DCX.DmmMode`.

AP.DCX.DmmRangeAuto ❶❷ Property

Syntax **`AP.DCX.DmmRangeAuto`**

Data Type Boolean

True	Auto range
False	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 `AP.DCX.DmmRange`

❶ **Example**

```
Sub Main
   AP.File.OpenTest "DCX1.AT1"      'Open test.
   AP.DCX.DmmRangeAuto  = 1   'set DMM input to auto _
      range.
   AP.DCX.DmmMode = 1      'set DMM mode to volts.
   AP.DCX.DmmRdgRate = 1   'set DMM reading rate to 25 _
      readings per second.
   AP.DCX.DmmScale = 2
   AP.DCX.DmmOffset = 1
   AP.DCX.DmmSettling(1, .20, "Vdc", 3, .03, 0) 'Set _
      settling parameters.
   AP.DCX.DmmTrig       'Trigger a new reading.
   Do
      Ready =  AP.DCX.DmmReady
   Loop Until Ready > 0 'Loop until new reading is _
         ready.
   Reading1 =  AP.DCX.DmmRdg("f(v)")   'Returns _
      a settled reading.

   NewLine$ = Chr(13)
   a$= "DMM Reading "+Left(Str$(Reading1),6)+"f(V)"
```

```
   AP.Prompt.Text = a$ + NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.DCX.DmmRdg ❶❷ Property

Syntax **AP.DCX.DmmRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	The following units are available VDC, V(f) for the AP.DCX.DmmMode command DCV mode and Ohms, and f(O) for the Ohms mode.

Description This command returns a settled reading for the DCX-127 Digital Multi meter(DMM) meter and zeros the ready count.

See Also AP.DCX.DmmMode, AP.DCX.DmmReady, AP.DCX.DmmSettling, AP.DCX.DmmTrig

Example See example for AP.DCX.DmmMode.

AP.DCX.DmmRdgRate ❶❷ Property

Syntax **AP.DCX.DmmRdgRate**

Data Type Integer

0	6 readings per second.
1	25 readings per second.

Description This command sets the DMM reading rate.

Example See example for AP.DCX.DmmMode.

AP.DCX.DmmReady

12 Property

Syntax `AP.DCX.DmmReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description 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 `AP.DCX.DmmRdg` or `AP.DCX.DmmTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.DCX.DmmRdg` command will be guaranteed to return quickly.

See Also `AP.DCX.DmmRdg, AP.DCX.DmmSettling, AP.DCX.DmmTrig`

Example See example for `AP.DCX.DmmMode`.

AP.DCX.DmmScale

12 Property

Syntax `AP.DCX.DmmScale`

Data Type Double

Description When f(V) (function of Volts) or f(O) (function of Ohms) units are selected for the DMM, APWIN software computes the value to display from the following formula:

```
display = (measurement + Offset) * 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 `AP.DCX.DmmOffset`

Example See example for `AP.DCX.DmmRangeAuto.`

AP.DCX.DmmSettling ❶❷ Method

Syntax **AP.DCX.DmmSettling(***tolerance#, floor#, floorunit$, points%, delay#, algorithm%***)**

Description This command sets the settling parameters for the AP.DCX.DmmRdg command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also AP.DCX.DmmRdg, AP.DCX.DmmReady, AP.DCX.DmmTrig

Example See example for AP.DCX.DmmMode.

AP.DCX.DmmTrig ❶❷ Method

Syntax **AP.DCX.DmmTrig**

Description Causes a restart of the reading cycle and zeros the ready count for the AP.DCX.DmmRdg command. The reading in progress is aborted.

See Also AP.DCX.DmmRdg, AP.DCX.DmmReady, AP.DCX.DmmSettling

Example See example for AP.DCX.DmmMode.

AP.DCX.PortAOutput ❶❷ Property

Syntax **AP.DCX.PortAOutput**

Data Type Integer — 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.

Parameters

Part	Description
unit$	The following units are available: dec, hex, oct.

Description This command sets DCX-127 Port A 8-bit output value.

❶ **Example**

```
Sub Main
```

```
   AP.File.OpenTest "DCX1.AT1"      'Open test.
   AP.DCX.PortAOutput("Dec") = 17
   AP.DCX.PortBOutput("Hex") = 34
   AP.DCX.PortCOutput("Oct") = 68
End Sub
```

AP.DCX.PortBOutput

❶❷ Property

Syntax `AP.DCX.PortBOutput`

Data Type Integer — 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.

Parameters

Part	Description
unit$	The following units are available: dec, hex, oct.

Description This command sets DCX-127 Port B 8-bit output value.

Example See example for `AP.DCX.PortAOutput`.

AP.DCX.PortCOutput

❶❷ Property

Syntax `AP.DCX.PortCOutput`

Data Type Integer — 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.

Parameters

Part	Description
unit$	The following units are available: dec, hex, oct.

Description This command sets DCX-127 Port C 8-bit output value.

Example See example for `AP.DCX.PortAOutput`.

User Notes

User Notes

User Notes

Digital Generator

AP.DGen.AmplRatio ❷ Property

Syntax **AP.DGen.AmplRatio(*unit$*)**

Data Type Double Valid settings are 0.00001% to 100%

Parameters

Name	Description
unit$	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 amplitude ratio to be used between the Frequency and the IM Frequency for the Digital SMPTE waveform selection.

See Also AP.DGen.Freq, AP.DGen.IMFreq

Example

```
Const SMPTE As Integer = 2
Const ANALYZER As Integer = 1
Const NOT_READY As Integer = 0
Const NONE As Integer = 0
Const FLAT As Integer = 2
Const EXPONENTIAL As Integer = 3

Sub Main
   AP.Application.NewTest  'Reset panels
   AP.DGen.Wfm IMD,SMPTE   'Put DGen in IM Mode (SMPTE).
   AP.DGen.Freq ("Hz")  = 2000'Set sine wave frequency.
   AP.DGen.IMFreq ("Hz") = 80 'Set IM freq to 80 Hz.
   AP.S2DSP.Program = ANALYZER  'Digital Domain Audio _
      Analyzer.
   AP.S2DSP.Analyzer.FuncSettling 1.0, 1e-3, "V", 3, _
      30e-3, EXPONENTIAL
   AP.DGen.Output = True               'Turn on output.
   For ratio = 51.0 To 1.00 Step -10   'Increment ratio.
      AP.DGen.AmplRatio("%") = ratio
      AP.S2DSP.Analyzer.FuncTrig
      While AP.S2DSP.Analyzer.FuncReady = NOT_READY
      Wend
```

```
            msg = msg & "Reading("&ratio&"%) = " & _
               AP.S2DSP.Analyzer.FuncRdg ("V") & Chr(13)
         Next
         AP.Prompt.Text = msg
         AP.Prompt.ShowWithContinue
         Stop
      End Sub
```

AP.DGen.BurstInterval ❷ Property

Syntax **AP.DGen.BurstInterval(*unit$*)**

Data Type Double 2 - 65535

Parameters

Name	Description
unit$	Cycles only.

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 AP.DGen.Wfm, AP.DGen.BurstLevel, AP.DGen.BurstOnTime

Example

```
Sub Main
   System = AP.Application.SysType
   if System = "1" Then Debug.Print"System Two test only.
   AP.Application.NewTest  'Reset panels
   AP.DGen.Wfm(0,1)
   AP.DGen.Output = 1
   AP.DGen.BurstInterval("Cycles") = 10
   AP.DGen.BurstOnTime("Cycles") = 5
   AP.DGen.BurstLevel("dB") = -40
   Interval = AP.DGen.BurstInterval("Cycles")
   Ontime = AP.DGen.BurstOnTime("Cycles")
   Level = AP.DGen.BurstLevel("%")
```

```
   Debug.Print "Burst Interval =";Interval;" cycles."
   Debug.Print "Burst ON time =";Ontime;" cycles."
   Debug.Print "Burst OFF time low level =";Level;" %."
End Sub
```

Example Output

```
Burst Interval = 10 cycles.
Burst ON time = 5 cycles.
Burst OFF time low level = 1 %.
```

AP.DGen.BurstLevel ❷ Property

Syntax `AP.DGen.BurstLevel(`*unit$*`)`

Data Type Double — Level of signal during burst off time. (0 - -80.25dB)

Parameters

Name	Description
unit$	The following units are available X/Y, dB, %, PPM.

Description 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).

See Also `AP.DGen.Wfm, AP.DGen.BurstInterval, AP.DGen.BurstOnTime`

Example See example for `AP.DGen.BurstInterval`.

AP.DGen.BurstOnTime ❷ Property

Syntax `AP.DGen.BurstOnTime(`*unit$*`)`

Data Type Double — From 1 to `AP.Gen.BurstInterval-1`.

Parameters

Name	Description
unit$	Cycles only.

Description 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.

See Also `AP.DGen.Wfm, AP.DGen.BurstInterval, AP.DGen.BurstLevel`

Example See example for `AP.DGen.BurstInterval`.

AP.DGen.ChAAmpl ❷ Property

Syntax **`AP.DGen.ChAAmpl(`*`unit$`*`)`**

Data Type Double — Valid amplitude settings are 0.0 to 100 %FS.

Parameters

Name	Description
`unit$`	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 channel A amplitude.

See Also `AP.DGen.ChBAmpl, AP.DGen.RefVFS`

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.DGen.Wfm(0,3)        'Stereo Sine.
   With AP.DGen
      .RefVFS("V") = 2     'Set Volts/FS ref = 5 volts.
      .RefFreq("Hz") = 10e3   'Set freq ref = 10 kHz.
      .RefdBr("FFS") = 0.5 'Set dBr ref = 2.5 volts.
      .ChAFreq("%Hz") = 50 'Set Ch A sinewave frequency.
      .ChBFreq("%Hz") = 75 'Set Ch B sinewave frequency.
      .ChAAmpl("dBr") = 0.0   'Set Ch A = 0.5 FFS.
      .ChBAmpl("dBr") = 0.0   'Set Ch B = 0.5 FFS.
      .ChAInvert = False   'Make sure Ch A invert is OFF.
      .ChBInvert = False   'Make sure Ch B invert is OFF.
      .ChBTrackA = False   'Make sure Ch B Track _
         Ch A is off.
      .ChAOutput = True    'Turn Ch A output ON.
      .ChBOutput = True    'Turn Ch A output ON.
      .OutDitherType = 0   'Triagular Dither.
```

```
      .Output = True 'Turn main output on (mute off).
   End With

   AP.S2Dsp.Program = 1    'Sellect DSP Audio Analyzer
   AP.S2Dio.InFormat = 3   'Generator monitor

   With AP.S2Dsp.Analyzer
      Do While (.ChALevelReady = False) Or _
         (.ChBLevelReady = False) Or _
         (.ChAFreqReady = False) Or _
         (.ChBFreqReady = False)
      Loop
      msg = "Ch A Level = " & Format(.ChALevelRdg("V"), _
         "#.00") & Chr(13)
      msg = msg & "Ch B Level = " & _
         Format(.ChBLevelRdg("V"), "#.00") & Chr(13)
      msg = msg & "Ch A Freq = " & _
         Format(.ChAFreqRdg("Hz"), "#.00") & Chr(13)
      msg = msg & "Ch B Freq = " & _
         Format(.ChBFreqRdg("Hz"), "#.00")
   End With
   AP.Prompt.Text = msg
   AP.Prompt.ShowWithContinue
   Stop
End Sub
```

AP.DGen.ChAEqAmpl

❷ Property

Syntax **AP.DGen.ChAEqAmpl(*unit$*)**

Data Type Double — Valid amplitude settings are 0.0 to 100 %FS.

Parameters

Name	Description
unit$	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 channel A post Eq amplitude.

See Also AP.DGen.ChBEqAmpl, AP.DGen.EqCurve

Example

```
Sub Main
   AP.Application.NewTest
   AP.Application.PanelClose apbPanelAnlrSmall
   AP.Application.PanelClose apbPanelAnalogGenSmall
   AP.Application.PanelOpen apbPanelDigitalGenLarge
   AP.DGen.ChAAmpl("dBFS") = -1.0
   AP.DGen.ChBAmpl("dBFS") = -1.0
   AP.DGen.EqCurve("75us-de.adq", 1)   'Load EQ file
   AP.DGen.Wfm 0, 6            'Select EQ Sine waveform
   AP.DGen.ChAEqAmpl("dBFS") = -1.0
   AP.DGen.ChBEqAmpl("dBFS") = -1.0
   AP.DGen.Output = True          'Generator Output On

   AP.S2Dsp.Program = 1           'Load DSP program
   AP.Application.PanelOpen apbPanelDSPLarge

   AP.Sweep.Data1.Id = 6014
   AP.Sweep.Data1.Top("dBFS") = 0.0
   AP.Sweep.Data1.Bottom("dBFS") = -25.0
   AP.Sweep.Source1.Id = 5102
   AP.Sweep.Stereo = True         'Stereo Sweep
   AP.Sweep.Start                 'Start Sweep
End Sub
```

AP.DGen.ChAFreq

❷ Property

Syntax **AP.DGen.ChAFreq(*unit$*)**

Data Type Double — Valid frequency range for each channel is 10 - 22.5 kHz.

Parameters

Name	Description
unit$	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 Digital Generator channel A frequency to be used when the Digital Generator waveform type is set to Sine Stereo.

See Also AP.DGen.

Example See example for AP.DGen.ChAAmpl.

AP.DGen.ChAInvert ❷ Property

Syntax **AP.DGen.ChAInvert**

Data Type Boolean

True	Invert channel A output.
False	Normal non-inverting output.

Description This command sets output A to normal polarity or inverted polarity (180 degrees out of phase).

See Also AP.DGen.ChBInvert

Example See example for AP.DGen.ChAAmpl.

AP.DGen.ChAOutput ❷ Property

Syntax **AP.DGen.ChAOutput**

Data Type Boolean

True	ON.
False	OFF.

Description This command sets the Digital Generator Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also AP.DGen.ChBOutput

Example See example for AP.DGen.ChAAmpl.

AP.DGen.ChBAmpl ❷ Property

Syntax **AP.DGen.ChBAmpl(*unit$*)**

Data Type Double — Valid amplitude settings are 0.0 to 100 %FS.

Parameters

Name	Description
unit$	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 channel B amplitude.

See Also AP.DGen.ChAAmpl

Example See example for AP.DGen.ChAAmpl.

AP.DGen.ChBEqAmpl ❷ Property

Syntax **AP.Gen.ChBEqAmpl(*unit$*)**

Data Type Double — Valid amplitude settings are 0.0 to 100 %FS.

Parameters

Name	Description
unit$	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 channel B post Eq amplitude.

See Also AP.DGen.ChAEqAmpl, AP.DGen.EqCurve

Example See example for AP.DGen.ChAEqAmpl.

AP.DGen.ChBFreq ❷ Property

Syntax **AP.DGen.Freq(*unit$*)**

Data Type Double

Valid frequency settings for the Hz unit and sine waveform are 10 - 22.5kHz for the 48kHz sample rate.

Parameters

Name	Description
unit$	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 Digital Generator channel B frequency when the waveform type is set to Sine Stereo.

See Also `AP.DGen.`

Example See example for `AP.DGen.ChAAmpl`.

AP.DGen.ChBInvert ❷ Property

Syntax **`AP.DGen.ChBInvert`**

Data Type Boolean

True	Invert channel B output.
False	Normal non-inverting output.

Description This command sets output B to normal polarity or inverted polarity (180 degrees out of phase with normal polarity).

See Also `AP.DGen.ChAInvert`

Example See example for `AP.DGen.ChAAmpl`.

AP.DGen.ChBOutput ❷ Property

Syntax **`AP.DGen.ChBOutput`**

Data Type Boolean

True	On
False	Off

Description This command sets the Digital Generator output B to ON or OFF.

The command returns a TRUE if the output is ON and a FALSE if the output is OFF.

See Also `AP.DGen.ChAOutput`

Example See example for `AP.DGen.ChAAmpl`.

AP.DGen.ChBTrackA ❷ Property

Syntax **`AP.DGen.ChBTrackA`**

Data Type 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 `AP.DGen.ChAAmpl, AP.DGen.ChBAmpl`

Example See example for `AP.DGen.ChAAmpl`.

AP.DGen.DitherType ❷ Property

Syntax **`AP.DGen.DitherType`**

Data Type Integer

`0`	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.
`1`	Rectangular: probability function dither provides the best signal to noise due to its one-half LSB amplitude, but suffers from modulation noise effects.
`2`	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.
	3	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 `AP.S2Dio.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 `AP.S2Dio.OutResolution` command.

Example See example for `AP.DGen.ChAAmpl`.

AP.DGen.DualAmplRatio ❷ Property

Syntax **`AP.DGen.DualAmplRatio(`***`unit$`***`)`**

Data Type Double — Valid settings are 0.00001% to 100%

Parameters

Name	Description
`unit$`	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 `AP.DGen.Freq, AP.DGen.ChAFreq, AP.DGen.ChBFreq`

AP.DGen.EqCurve ❷ Method

Syntax **`AP.DGen.EqCurve(`***`filename$, column%`***`)`**

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN Eq file (.adq).
column%	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 open successfull.
False	File open failed.

Description This command attaches a Eq file to the Digital Generator. Values in the file will be used as multiply factors in czlculating the Digital Generator Amplitude values.

See Also `AP.DGen.ChAEqAmpl, AP.DGen.ChBEqAmpl`

Example See example for `AP.DGen.ChAEqAmpl`.

AP.DGen.Freq ❷ Property

Syntax **`AP.DGen.Freq(`***`unit$`***`)`**

Data Type Double — Valid frequency settings for the Hz unit and sine waveform are 10 - 22.5kHz for the 48kHz sample rate.

Parameters

Name	Description
unit$	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 Digital Generator main frequency for the Sine waveforms.

AP.DGen.IMCenterFreq ❷ Property

Syntax `AP.DGen.IMCenterFreq(unit$)`

Data Type Double

Parameters

Name	Description
unit$	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 Digital Generator IMD Center Frequency. The frequency passed is rounded to the closest available value.

Set the Digital Generator waveform to an IMD CCIF before calling this command.

See Also `AP.DGen.Wfm, AP.DGen.IMFreq`

AP.DGen.IMFreq ❷ Property

Syntax `AP.DGen.IMFreq(unit$)`

Data Type 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.

Parameters

Name	Description
unit$	String that designates the desired unit. The following unit is valid for this command. Hz

Description 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 or CCIF) before calling this command.

See Also `AP.DGen.Wfm, AP.DGen.IMCenterFreq`

AP.DGen.IMHighFreq ❷ Property

Syntax `AP.DGen.IMHighFreq(unit$)`

Data Type Double

Parameters

Name	Description
unit$	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 before calling this command.

See Also `AP.DGen.Wfm, AP.DGen.IMFreq`

AP.DGen.Offset ❷ Property

Syntax `AP.Gen.Offset(unit$)`

Data Type DoubleParameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits, Vrms, Vp, Vpp, dBu, dBV, dBr, dBrInv

Description This command sets the Digital Generator Sine + Offset waveform Offset value.

See Also `AP.DGen.Wfm`

AP.DGen.Output ❷ Property

Syntax `AP.Gen.Output`

Data Type Boolean

True	On
False	Off

Description This command sets the Digital Generator channel A and B outputs to ON or OFF if they have been individually enabled by the `AP.DGen.ChAOutput` and `AP.DGen.ChBOutput` commands.

See Also `AP.DGen.ChAOutput, AP.DGen.ChBOutput`

Example See example for `AP.DGen.ChAAmpl`.

AP.DGen.Phase ❷ Property

Syntax **`AP.DGen.Phase(`***`unit$`***`)`**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: deg

Description This command sets the Digital Generator Phase value.

Set the Digital Generator waveform to Sine Var Phase before calling this command.

See Also `AP.DGen.Wfm`

Example

```
Sub Main
   AP.Application.NewTest
   AP.Application.PanelClose apbPanelAnalogGenSmall
   AP.Application.PanelOpen apbPanelDigitalGenSmall
   AP.DGen.Wfm 0, 2
   AP.DGen.Phase("deg") = 90.000000
   AP.DGen.Output = True
   'Send digital signal through D/A converter.
   AP.Sweep.Data1.Id = 5905
   AP.Sweep.SinglePoint = True
   AP.Sweep.Start
```

```
   Debug.Print “Channel B is ” &
Format(AP.Data.Value(0,1,0,"deg"),"##.000") & “ deg
relative to channel A”
End Sub
```

Example Output

```
Channel B is 89.994 deg relative to channel A.
```

AP.DGen.RefdBr ❷ Property

Syntax **AP.DGen.RefdBr(*unit$*)**

Data Type Double — Amplitude value.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits, V, Vp, Vpp, dBu, dBV

Description This command sets the zero dBr value for the Digital Generator dBr unit.

See Also AP.DGen.ChAAmpl, AP.DGen.ChBAmpl, AP.DGen.RefVFS

Example See example for AP.DGen.ChAAmpl.

AP.DGen.RefFreq ❷ Property

Syntax **AP.DGen.RefFreq(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	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 `AP.DGen.Freq, AP.DGen.ChAFreq, AP.DGen.ChBFreq`

Example See example for `AP.DGen.ChAAmpl`.

AP.DGen.RefVFS ❷ Property

Syntax **`AP.Gen.RefVFS(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
`unit$`	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 `AP.DGen.ChAAmpl`.

AP.DGen.StepRate ❷ Property

Syntax **`AP.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 `AP.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 `AP.DGen.Wfm`

AP.DGen.Wfm

❷ Property

Syntax **AP.DGen.Wfm** *primary%, secondary*

Data Type Integer

Parameters

Name	Description
primary%	This parameter defines the basic waveform type.
secondary	This parameter defines the basic waveform modifier.

Primary	Secondary	Description
0		Sine
	0	Normal
	1	Burst
	2	Var Phase
	3	Stereo
	4	Dual
	5	Sine + Offset
	6	EQ Sine
	6	Shaped Burst
1		Square.
2		IMD
	0	SMPTE
	1	CCIF
	2	DIM
3		Noise
	0	Pink
	1	White
	2	Burst USASI
4		Special
	0	Monotonicity
	1	J-Test
	2	Polarity

	3	Walking Ones
	4	Walking Zeros
	5	Constant Value
5		MLS
	0	Pink #1
	1	Pink #2
	2	Pink #3
	3	Pink #4
	4	White #1
	5	White #2
	6	White #3
	7	White #4
6		Arb Wfm

Description This command sets the Digital Generator waveform. The table above shows the possible settings for the `AP.DGen.Wfm` command.

Example See example for `AP.DGen.WfmName`.

AP.DGen.WfmName

❷ **Method**

Syntax **`AP.DGen.WfmName =`** *`filename$`*

Parameters

Name	Description
`filename$`	Any valid DOS path and file name. The file must be an APWIN waveform file (.agm or .ags).

Result Boolean

`True`	File open successfull.
`False`	File open failed.

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 `AP.DGen.Wfm`

Example

```
Sub Main
   AP.Application.NewTest
   AP.Application.PanelClose apbPanelAnalogGenSmall
   AP.Application.PanelClose apbPanelAnlrSmall
   AP.Application.PanelOpen apbPanelDigitalGenSmall
   AP.Application.PanelOpen apbPanelDSPSmall
   'Load Digital Analyzer (Multitone Audio Analyzer)
   AP.S2Dsp.Program = 4
   AP.Application.PanelOpen apbPanelDigIOSmall
   'Select Gen Mon on the Digital I/O panel to route the
   ' Digital generator directly To the Digital Analyzer.
   AP.S2Dio.InFormat = 3
   AP.DGen.Wfm 6           'Select arbitrary waveform
   AP.DGen.WfmName = "C:\Apwin\Waveform\Iso31.agm"
   AP.DGen.Output = True   'Digital Generator Output ON
   'Set up Sweep panel to display test data.
   AP.Application.PanelOpen apbPanelSweepSmall
   AP.Sweep.Data1.Id = 6309
   AP.Sweep.Data1.Top("dBFS") = 0.000000

   AP.Sweep.Source1.Id = 5621
   AP.Sweep.Source1.Steps = 200
   AP.Sweep.Stereo = True

   AP.Sweep.Start                'Run Test
   AP.Graph.OptimizeLeft
End Sub
```

User Notes

User Notes

User Notes

File

AP.File.AppendData ❶ Method

Syntax `AP.File.ImportDOSS1Test(filename$)`

Parameters

Name	Description
filename$	Any valid DOS filename and extension.

Result Boolean

True	File Data Append successfull.
False	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.

See Also `AP.File.AppendTest`

AP.File.ExportASCIIData ❶❷ Method

Syntax `AP.File.ExportASCIIData(filename$)`

Parameters

Name	Description
filename$	Any valid DOS filename and extension.

Result Boolean

True	File export successfull.
False	File export failed.

Description This command saves the measurement data in memory to a coma delimited ASCII text file.

See Also `AP.File.ImportASCIIData`

Example

```
Sub Main
'Smooth Data from ASCII Data file
   AP.Application.NewTest
   AP.Gen.Output = True
   AP.Anlr.ChAInput = 1
'Load ASCII data file
   AP.File.ImportASCIIData("TEMP.adx")
   AP.Application.PanelOpen apbPanelGraph
   AP.Sweep.Data1.LogLin = 1
   AP.Graph.OptimizeLeft
   If AP.Sweep.Data1.Id <> 5049 Then _
      AP.Compute.Smooth.Data(1) = True
   If AP.Sweep.Data2.Id <> 5049 Then _
      AP.Compute.Smooth.Data(2) = True
   If AP.Sweep.Data3.Id <> 5049 Then _
      AP.Compute.Smooth.Data(3) = True
   If AP.Sweep.Data4.Id <> 5049 Then _
      AP.Compute.Smooth.Data(4) = True
   If AP.Sweep.Data5.Id <> 5049 Then _
      AP.Compute.Smooth.Data(5) = True
   If AP.Sweep.Data6.Id <> 5049 Then _
      AP.Compute.Smooth.Data(6) = True
   AP.Compute.Smooth.Auto = True
   AP.Compute.Smooth.Apply
'Export ASCII data file
   AP.File.ExportASCIIData("TEMP.adx")
End Sub
```

AP.File.ExportGraphic

❶❷ Method

Syntax **AP.File.ExportGraphic(***filename$, type%***)**

Parameters

Name	Description
filename$	Any valid DOS filename and extension.
type%	0 = Windows Meta File (.WMF). 1 = Windows Extended Meta File (.EMF).

Result Boolean

True	File export successfull.
False	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
   AP.Application.NewTest
   AP.Gen.Output = True
   AP.Anlr.ChAInput = 2
   AP.Anlr.FuncFilterLP = 0
   AP.Anlr.FuncFilterHP = 3
   AP.Sweep.Data1.Id = 5906
   AP.Sweep.Source1.Start("Hz") = 50000
   AP.Sweep.Start

   Kill "C:\GRAPH.EMF"

   ' Export Windows Meta File
   blnExport = AP.File.ExportGraphic("C:\GRAPH.EMF", 1)
   If blnExport = True Then End

   Dim MSWord As Object
   Set MSWord = CreateObject("Word.Basic")' Start Word
   MSWord.AppShow    ' Word is invisible on startup.
                     ' Set to visible
   MSWord.FileOpen Name:= "C:\GENERIC.DOC"
   MSWord.EditFind "Graph"    ' Search for string
   ' Import Windows Meta File Graph
   MSWord.InsertPicture "C:\GRAPH.EMF"
   MSWord.FilePrint           ' Print Doc from MS Word
   Wait 10
   MSWord.FileCloseAll 2      ' Close all open files
   MSWord.AppClose            ' Close MS Word
End Sub
```

AP.File.ImportASCIIData ❶❷ Method

Syntax `AP.File.ImportASCIIData(`*`filename$`*`)`

Parameters

Name	Description
filename$	Any valid DOS filename and extension.

Result Boolean

True	File import successfull.
False	File import failed.

Description This command loads into memory the designated ASCII data file. This comand only loads files that have been exported from APWIN or conform to the APWIN ASCII data file format.

See Also `AP.File.ExportASCIIData`

Example See example for `AP.File.ExportASCIIData`.

AP.File.ImportDOSS1Test ❶ Method

Syntax `AP.File.ImportDOSS1Test(`*`filename$`*`[, `*`ignorefile`*`])`

Parameters

Name	Description
filename$	Any valid DOS filename and extension.
ignorefile	True = Remove attached files. False = Keep attached files.

Result Boolean

True	File import successfull.
False	File import failed.

Description This command translates S1.EXE test files to the APWIN System One configuration only.

See Also `AP.Application.SysType`

AP.File.OpenData

❶❷ Method

Syntax **AP.File.OpenData(*filename$*)**

Parameters

Name	Description
filename$	Any valid DOS filename and extension.

Result Boolean

True	File open successfull.
False	File open failed.

Description This command loads the designated data file.

❶ **Example**

```
Sub Main
   OpenResult = AP.File.OpenTest(“FRQ-RESP.AT1")
   If OpenResult = False Then Call Open_Failed
   OpenResult = AP.File.OpenData(“FRQ-RESP.DAT”)
   If OpenResult = False Then Call Open_Failed
   AP.Data.UpdateDisplay 0
   Wait 5
   OpenResult = AP.File.OpenTest(“THD-FRQ.AT1")
   If OpenResult = False Then Call Open_Failed
   OpenResult = AP.File.OpenData(“THD-FRQ.DAT”)
   If OpenResult = False Then Call Open_Failed
   AP.Data.UpdateDisplay 0
   Wait 5
   OpenResult = AP.File.OpenTest(“RESIDNOI.AT1")
   If OpenResult = False Then Call Open_Failed
   OpenResult = AP.File.OpenData(“RESIDNOI.DAT”)
   If OpenResult = False Then Call Open_Failed
   AP.Data.UpdateDisplay 0
   Wait 5
End Sub

Sub Open_Failed
   Debug.Print"File Open FAILED."
   End
End Sub
```

AP.File.OpenMacro ❶❷ Method

Syntax **AP.File.OpenMacro(*filename$*)**

Parameters

Name	Description
filename$	Any valid DOS filename and extension.

Result Boolean

True	Not applicable.
False	File open failed.

Description This command loads the designated file into the macro editor and automatically runs the macro.

Example

```
'Visual Basic example
Private Sub Form_Load()
   Dim AP As Object
   'Create OLE link to APWIN.
   Set AP = CreateObject("APWIN.Application")
   AP.Application.Visible = True ' Make APWIN visible

   'Place your code here

   'Run an APWIN Macro and wait for it to finish
   AP.File.OpenMacro "C:\BUSY.APB"
   While AP.Macro.IsRunning = True
   Wend

   'Change Visual Basic directory to APWIN Working _
      Directory.
   ChDir AP.Application.MacroDir

   'Place your code here

   AP.Application.Quit           'Quit APWIN
   End
End Sub
```

AP.File.OpenTest

❶❷ Method

Syntax **AP.File.OpenTest(***filename$***)**

Parameters

Name	Description
filename$	Any valid DOS filename and extension.

Result Boolean

True	File open successfull.
False	File open failed.

Description This command loads the designated test file.

❶ **Example**

```
Sub Main
   OpenResult = AP.File.OpenTest(“FRQ-RESP.AT1")
   If OpenResult = False Then Call Open_Failed
   AP.Sweep.Start
   SaveResult = AP.File.SaveDataAs(“FRQ-RESP.DAT”)
   If SaveResult = False Then Call Save_Failed

   OpenResult = AP.File.OpenTest(“THD-FRQ.AT1")
   If OpenResult = False Then Call Open_Failed
   AP.Sweep.Start
   SaveResult = AP.File.SaveDataAs(“THD-FRQ.DAT”)
   If SaveResult = False Then Call Save_Failed

   OpenResult = AP.File.OpenTest(“RESIDNOI.AT1")
   If OpenResult = False Then Call Open_Failed
   AP.Sweep.Start
   SaveResult = AP.File.SaveDataAs(“RESIDNOI.DAT”)
   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
```

AP.File.OpenWfm

❶❷ Method

Syntax **AP.File.OpenWfm(**filename$, *option1%, option2%*)

Parameters

Name	Description
filename$	Any valid DOS filename and extension.
option1%	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
option2%	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 acqusition (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

❷ **Example**

```
Sub Main
   AP.File.OpenTest("FFTSAVE.AT2")
   OpenResult = AP.File.OpenWfm("TEMP.AAS", 1,2)
   If OpenResult = False Then Call Open_Failed
   AP.Sweep.Reprocess
End Sub

Sub Open_Failed
   Debug.Print"Test Open FAILED."
   End
End Sub
```

AP.File.SaveAll ❶❷ Method

Syntax **AP.File.SaveAll**

Parameters None

Result Boolean

True	File save successfull.
False	File save failed.

Description This command saves the current test and all macros loaded in the macro editor.

AP.File.SaveDataAs ❶❷ Method

Syntax **AP.SaveDataAs(***filename$***)**

Parameters

Name	Description
filename$	Any valid DOS filename and extension.

Result Boolean

True	File save successfull.
False	File save failed.

Description This command saves the measurement data in memory to the designated file.

Example See example for `AP.File.OpenTest`.

AP.File.SaveMacro ❶❷ Method

Syntax **`AP.File.SaveMacro`**

Parameters None

Result Boolean

True	File save successfull.
False	File save failed.

Description This command saves the macro currently selected in the macro editor.

AP.File.SaveMacroAs ❶❷ Method

Syntax **`AP.File.SaveMacroAs(`***`filename$`***`)`**

Parameters

Name	Description
filename$	Any valid DOS filename and extension.

Result Boolean

True	File save successfull.
False	File save failed.

Description This command saves the macro currently in the macro editor to the designated file.

Example

```
Sub Main
   Result = AP.File.SaveMacroAs("TEMP.APB")
```

```
   If Result = False Then
      Debug.Print “Macro NOT saved.”
   Else
      Debug.Print “Macro saved.”
   End If
End Sub
```

Example Output

```
Macro saved.
```

AP.File.SaveTest ❶❷ Method

Syntax **AP.File.SaveTest**

Parameters None

Result Boolean

True	File save successfull.
False	File save failed.

Description This command saves the current test.

❶ Example

```
Sub Main
   AP.File.OpenTest “FRQ-RESP.AT1"  ‘Open frequency
response test.
   AP.Sweep.Start                   ‘Start sweep.
   If AP.File.SaveTest = False Then GoTo Quit ‘Save Test
   AP.File.OpenTest “THD-FRQ.AT1"   ‘Open total harmonic
distortion + noise test.
   AP.Sweep.Start                   ‘Start sweep.
   If AP.File.SaveTest = False Then GoTo Quit ‘Save Test
   AP.File.OpenTest “RESIDNOI.AT1"  ‘Open residual noise
test.
   AP.Sweep.Start                   ‘Start sweep.
   If AP.File.SaveTest = False Then GoTo Quit ‘Save Test
   End
   Quit:
   Debug.Print “Test Save FAILED”
End Sub
```

AP.File.SaveTestAs

❶❷ Method

Syntax **AP.File.SaveTestAs(*filename$*)**

Parameters

Name	Description
filename$	Any valid DOS filename and extension.

Result Boolean

True	File save successfull.
False	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.

❶ **Example**

```
Sub Main
   OpenResult = AP.File.OpenTest(“FRQ-RESP.AT1")
   If OpenResult = False Then Call Open_Failed
   AP.Sweep.Start
   SaveResult = AP.File.SaveTestAs(“FRQ-RESP.AT1")
   If SaveResult = False Then Call Save_Failed

   OpenResult = AP.File.OpenTest(“THD-FRQ.AT1")
   If OpenResult = False Then Call Open_Failed
   AP.Sweep.Start
   SaveResult = AP.File.SaveTestAs(“THD-FRQ.AT1")
   If SaveResult = False Then Call Save_Failed

   OpenResult = AP.File.OpenTest(“RESIDNOI.AT1")
   If OpenResult = False Then Call Open_Failed
   AP.Sweep.Start
   SaveResult = AP.File.SaveTestAs(“RESIDNOI.AT1")
   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
```


AP.File.SaveWfm

❶❷ Method

Obsolete Obsolete command not recomended for new design. This command has been renamed to `AP.File.SaveWfmAs`.

AP.File.SaveWfmAs

❶❷ Method

Syntax **`AP.File.SaveWfmAs(`***`filename$,`* `option1%,` ***`option2%`*****`)`**

Parameters

Name	Description
`filename$`	Any valid DOS filename and extension.
`option1%`	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
`option2%`	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 acqusition (F9). Selecting the acquisition buffer causes the complete acquired signal to be saved to a disk file for later download (via the `AP.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

❷ **Example**

```
Sub Main
   AP.File.OpenTest “FFTSAVE.AT2"
   AP.Sweep.Start
   AP.File.SaveWfmAs “TEMP.AAS”, 1,2
End Sub
```

User Notes

User Notes

User Notes

Analog Generator

AP.Gen.Ampl ❶❷ Property

Syntax **AP.Gen.Ampl(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: Vrms, Vp, Vpp, dBu, dBV, dBr, dBm, W, dBrInv

Description This command sets the Analog Generator channel A and B amplitude.

Example

```
Sub Main
   Dim reading(1 To 31)    'Dimension array.
   ndx = 1                 'Array index.
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = True    'Turn output ON.
   AP.Anlr.ChAInput = 1    'Select GENMON internal _
      connection.
   AP.Gen.Ampl("Vrms") = 5    'Set output level to 5V.
   'Sweep 20 Hz to 20 kHz in 30 linear steps.
   For NewFreq = 20 To 20e3 Step (20e3 - 20)/30
      AP.Gen.Freq ("Hz") = NewFreq
      'Measure amplitude from DUT.
      reading(ndx) = AP.Anlr.FuncRdg ("V")
      ndx = ndx + 1
   Next
End Sub
```

AP.Gen.AmplRatio ❷ Property

Syntax **AP.Gen.AmplRatio(*unit$*)**

Data Type Double — Valid settings are 0.00001% to 100%

Parameters

Name	Description
unit$	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 (D/A) Dual waveform selection.

See Also AP.Gen.ChAFreq, AP.Gen.ChBFreq

AP.Gen.BurstInterval ❶❷ Property

Syntax **AP.Gen.BurstInterval(*unit$*)**

Data Type Double — 2 - 65535

Parameters

Name	Description
unit$	Cycles only.

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 AP.Gen.Wfm, AP.Gen.BurstLevel, AP.Gen.BurstOnTime

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.Wfm(0,1)
   AP.Gen.Output = 1
   AP.Gen.BurstInterval("Cycles") = 10
```

```
    AP.Gen.BurstOnTime("Cycles") = 5
    AP.Gen.BurstLevel("dB") = -40
    Interval = AP.Gen.BurstInterval("Cycles")
    Ontime = AP.Gen.BurstOnTime("Cycles")
    Level = AP.Gen.BurstLevel("%")
    Debug.Print "Burst Interval =";Interval;" cycles."
    Debug.Print "Burst ON time =";Ontime;" cycles."
    Debug.Print "Burst OFF time low level =";Level;" %."
End Sub
```

Example Output

```
Burst Interval = 10 cycles.
Burst ON time = 5 cycles.
Burst OFF time low level = 1 %.
```

AP.Gen.BurstLevel

❶❷ Property

Syntax **AP.Gen.BurstLevel(*unit$*)**

Data Type Double — Level of signal during burst off time. (0 - -80.25dB)

Parameters

Name	Description
unit$	The following units are available X/Y, dB, %, PPM.

Description 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).

See Also AP.Gen.Wfm, AP.Gen.BurstInterval, AP.Gen.BurstOnTime

Example See example for AP.Gen.BurstInterval.

AP.Gen.BurstOnTime

❶❷ Property

Syntax **AP.Gen.BurstOnTime(*unit$*)**

Data Type Double — From 1 to AP.Gen.BurstInterval - 1.

Parameters

Name	Description

unit$	Cycles only.

Description 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.

See Also `AP.Gen.Wfm, AP.Gen.BurstInterval, AP.Gen.BurstLevel`

Example See example for `AP.Gen.BurstInterval`.

AP.Gen.ChAAmpl ❷ Property

Syntax **`AP.Gen.ChAAmpl(`***`unit$`***`)`**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: Vrms, Vp, Vpp, dBu, dBV, dBr, dBm, W

Description This command sets the Analog Generator channel A amplitude.

See Also `AP.Gen.ChBAmpl`

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.ChAOutput = 1
   AP.Gen.ChBOutput = 1
   AP.Gen.ChAAmpl("Vrms") = 1
   AP.Gen.ChBAmpl("Vrms") = 2
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 2
   AP.Anlr.ChBInput = 2
   AP.Anlr.ChALevelSettling(1, .000025, "V", 3, .03, 1)
   AP.Anlr.ChBLevelSettling(1, .000025, "V", 3, .03, 1)
   AP.Anlr.ChALevelTrig
   AP.Anlr.ChBLevelTrig
   Do
```

```
      ReadyA = AP.Anlr.ChALevelReady
      ReadyB = AP.Anlr.ChBLevelReady
   Loop Until ReadyA > 0 And ReadyB > 0
   ReadingA = AP.Anlr.ChALevelRdg("V")
   ReadingB = AP.Anlr.ChBLevelRdg("V")
   AP.Prompt.Text = "Level A amplitude =" & Format _
      (ReadingA,"#.0000") & " V" & Chr(13) & "Level B _
      amplitude =" & Format(ReadingB,"#.0000") & " V"
   AP.Prompt.ShowWithContinue
   Stop              'Wait of user to press continue.
End Sub
```

AP.Gen.ChAEqAmpl

❷ Property

Syntax **AP.Gen.ChAEqAmpl(*unit$*)**

Data Type Double — Valid amplitude settings are 0.0 to 100 %FS.

Parameters

Name	Description
unit$	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 channel A post Eq amplitude.

See Also AP.Gen.ChBEqAmpl, AP.Gen.EqCurve

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.EqCurve("75us-pre.adq", 1)   'Load EQ file
   AP.Gen.Wfm 0, 4            'Select EQ Sine waveform
   AP.Gen.ChAEqAmpl("dBV") = -10.0
   AP.Gen.ChBEqAmpl("dBV") = -10.0
   AP.Gen.Output = True       'Generator Output On

   AP.Sweep.Data1.Id = 5903
   AP.Sweep.Source1.Id = 5051
   AP.Sweep.Data1.Top("dBV") = 12.0
   AP.Sweep.Data1.Bottom("dBV") = -12.0
```

```
   AP.Sweep.Stereo = True
   AP.Sweep.Start
End Sub
```

AP.Gen.ChAFreq

❶❷ Property

Syntax **AP.Gen.ChAFreq(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	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 Analog Generator channel A frequency.

See Also AP.Gen.ChBFreq

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.Wfm 1, 2      'Select Stereo Sine waveform
   AP.Gen.ChAFreq("Hz") = 5000.0 'Set cha A frequency
   AP.Gen.ChBFreq("Hz") = 7000.0 'Set cha B frequency
   AP.Gen.ChBTrackA = False 'Set amplitude tracing OFF
   AP.Gen.ChAAmpl("dBV") = -0.0  'Set cha A amplitude
   AP.Gen.ChBAmpl("dBV") = -20.0 'Set cha B amplitude
   AP.Anlr.ChAInput = 2 'Select channel A Generator _
      Monitor input
   AP.Anlr.ChBInput = 2 'Select channel B Generator _
      Monitor input
   AP.Anlr.FuncMode = 9 'Select 2-Ch. Ratio measurement
   AP.Anlr.FuncInput = 1   'Measure B relative to A
   AP.Gen.Output = True    'Turn on generator output
   var = AP.Anlr.FuncRdg("dB")   'Return measurement
   Debug.Print "Channel B Amlitude is " & _
      Format(var,"##.0000") & " dB relative to A"
End Sub
```

AP.Gen.ChAInvert ❷ Property

Syntax **AP.Gen.ChAInvert**

Data Type Boolean

True	Invert channel A output.
False	Normal non-inverting output.

Description This command sets Analog Generator channel A output to normal polarity or inverted polarity (180 degrees out of phase).

See Also AP.Gen.ChBInvert

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.ChAInvert = False
   AP.Gen.ChBInvert = True
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 2
   AP.Anlr.ChBInput = 2
   AP.Anlr.PhaseSettling(0, .5, "deg", 3, .03, 1)
   AP.Anlr.PhaseTrig
   Do
      Ready = AP.Anlr.PhaseReady
   Loop Until Ready > 0
   Reading = AP.Anlr.PhaseRdg("deg")
   Debug.Print "Phase A to B = ";Format(Reading, _
      "#.0000");" deg"
End Sub
```

Example Output

```
Phase A to B = 180.0110 deg
```

AP.Gen.ChAOutput ❶❷ Property

Syntax **AP.Gen.ChAOutput**

Data Type Boolean

True	ON.
False	OFF.

Description This command sets the Analog Generator channel A output to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.Gen.ChBOutput`

Example See example for `AP.Gen.ChAAmpl`.

AP.Gen.ChBAmpl ❷ Property

Syntax **`AP.Gen.ChBAmpl(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
`unit$`	String that designates the desired unit. The following units are valid for this command: Vrms, Vp, Vpp, dBu, dBV, dBr, dBm, W

Description This command sets the generator channel B amplitude.

See Also `AP.Gen.ChAAmpl`

Example See example for `AP.Gen.ChAAmpl`.

AP.Gen.ChBEqAmpl ❷ Property

Syntax **`AP.Gen.ChBEqAmpl(`*`unit$`*`)`**

Data Type Double Valid amplitude settings are 0.0 to 100 %FS.

Parameters

Name	Description
`unit$`	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 channel B post Eq amplitude.

See Also `AP.Gen.ChAEqAmpl, AP.Gen.EqCurve`

Example See example for `AP.Gen.ChAEqAmpl`.

AP.Gen.ChBFreq ❶❷ Property

Syntax **`AP.Gen.ChBFreq(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
`unit$`	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 Analog Generator channel B frequency.

See Also `AP.Gen.ChAFreq`

AP.Gen.ChBInvert ❶❷ Property

Syntax **`AP.Gen.ChBInvert`**

Data Type Boolean

`True`	Invert channel B output.
`False`	Normal non-inverting output.

Description This command sets output B to normal polarity or inverted polarity (180 degrees out of phase with channel A normal polarity).

See Also `AP.Gen.ChAInvert (System Two only)`

Example See example for `AP.Gen.ChAInvert`.

AP.Gen.ChBOutput ❶❷ Property

Syntax `AP.Gen.ChBOutput`

Data Type Boolean

True	On
False	Off

Description This command sets output B to ON or OFF.

The command returns a TRUE if the output is ON and a FALSE if the output is OFF.

See Also `AP.Gen.ChAOutput`

Example See example for `AP.Gen.ChAAmpl`.

AP.Gen.ChBTrackA ❷ Property

Syntax `AP.Gen.ChBTrackA`

Data Type Boolean

True	ON, channel B amplitude tracks channel A amplitude.
False	OFF, channel B amplitude independent of channel A.

Description This command sets channel "B" amplitude to the same amplitude as set for channel "A".

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.ChAOutput = 1
   AP.Gen.ChBOutput = 1
   AP.Gen.ChAAmpl("Vrms") = 1
   AP.Gen.ChBTrackA = 1
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 2
   AP.Anlr.ChBInput = 2
   AP.Anlr.ChALevelSettling(1, .000025, "V", 3, .03, 1)
   AP.Anlr.ChBLevelSettling(1, .000025, "V", 3, .03, 1)
```

```
   AP.Anlr.ChALevelTrig
   AP.Anlr.ChBLevelTrig
   Do
      ReadyA = AP.Anlr.ChALevelReady
      ReadyB = AP.Anlr.ChBLevelReady
   Loop Until ReadyA > 0 And ReadyB > 0
   ReadingA = AP.Anlr.ChALevelRdg("V")
   ReadingB = AP.Anlr.ChBLevelRdg("V")
   Debug.Print "Level A amplitude = "; _
      Format(ReadingA,"#.0000");" V"
   Debug.Print "Level B amplitude = "; _
      Format(ReadingB,"#.0000");" V"
End Sub
```

Example Output

```
Level A amplitude = .9970 V
Level B amplitude = .9995 V
```


AP.Gen.Config

❶❷ Property

Syntax `AP.Gen.Config`

Data Type Integer

0	Bal - Float.
1	Bal - Gnd.
2	Unbal - Float.
3	Unbal - Gnd.
4	CMTST.

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 grounded or floating.

This command sets both outputs to a common mode test configuration.

See Also AP.Gen.Impedance

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.Config = 0 'Set output configuration to _
      balanced floating.
   AP.Gen.Impedance = 2    'Set generator output _
      impedance to 600 ohms.
   AP.Gen.Ampl("dBm") = 0
   AP.Gen.Output = 1
   AP.Anlr.ChBRangeAuto = 0   'Set input ranging to fixed.
   AP.Anlr.ChBRange("V") = 2.5 'Set input range to _
      2.5 Volts.
   AP.Anlr.ChBInput = 0 'Set anlr input to INPUT(XLR).
   AP.Anlr.ChBImpedance = 1      'Set Cha A input Z to _
      600 ohms.
   AP.Anlr.FuncInput = 1   'Set Function Meter Cha to B.
   AP.Anlr.FuncSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.FuncTrig
   Do
      Ready = AP.Anlr.FuncReady
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg("dBm")
   Debug.Print "Channel B Amplitude = ";Format(Reading, _
      "#.0000");" dBm"
   AP.Anlr.ChBRangeAuto = 1 'Set input ranging to auto.
End Sub
```

Example Output

```
Channel B Amplitude = -105.9976 dBm
```

AP.Gen.DualAmplRatio

❷ Property

Syntax **AP.Gen.DualAmplRatio(*unit$*)**

Data Type Double — Valid settings are 0.00001% to 100%

Parameters

Name	Description
unit$	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 IMD (D/A) SMPTE waveform selection.

See Also `AP.Gen.ChAFreq, AP.Gen.ChBFreq`

AP.Gen.EqAmpl ❶❷ Property

Syntax **`AP.Gen.EqAmpl(`***`unit$`***`)`**

Data Type Double — Valid amplitude settings are 0.0 to 100 %FS.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: Vrms, Vp, Vpp, dBu, dBV, dBr, dBm, W, dBrInv

Description This command sets the Analog Generator post Eq amplitude.

See Also `AP.Gen.EqCurve`

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.EqCurve("75us-pre.adq", 1)   'Load EQ file
   AP.Gen.Wfm 0, 4   'Select EQ Sine waveform
   AP.Gen.EqAmpl("dBV") = -10.0
   AP.Gen.Output = True 'Generator Output On
   AP.Sweep.Data1.Id = 5903
   AP.Sweep.Source1.Id = 5051
   AP.Sweep.Data1.Top("dBV") = 12.0
   AP.Sweep.Data1.Bottom("dBV") = -12.0
   AP.Sweep.Stereo = True
   AP.Sweep.Start
End Sub
```

AP.Gen.EqCurve ❶❷ Method

Syntax **`AP.Gen.EqCurve(`***`filename$, column%`***`)`**

Data Type Boolean

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN Eq file (.adq).
column%	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 `AP.Gen.ChAEqAmpl`.

AP.Gen.Freq

❶❷ Property

Syntax **`AP.Gen.Freq(`***`unit$`***`)`**

Data Type Double — Valid frequency settings for the Hz unit and sine waveform are 10 - 204775.

Parameters

Name	Description
unit$	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 Analog Generator frequency.

See Also `AP.Gen.FreqAccuracy`

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = 1
   AP.Gen.Freq("Hz") = 10000
   AP.Anlr.ChAInput = 1
```

```
   AP.Anlr.ChAFreqSettling(.5, .0002, “Hz”, 3, .03, 1)
   AP.Anlr.ChAFreqTrig
   Do
      Ready = AP.Anlr.ChAFreqReady
   Loop Until Ready > 0
   Reading = AP.Anlr.ChAFreqRdg(“Hz”)
   Debug.Print “Fast Frequency = ”;Format(Reading, _
      “#.0000");” Hz"
   AP.Gen.FreqAccuracy = 1       ‘Set Frequency to _
      High Accuracy.
   AP.Anlr.ChAFreqTrig
   Do
      Ready = AP.Anlr.ChAFreqReady
   Loop Until Ready > 0
   Reading = AP.Anlr.ChAFreqRdg(“Hz”)
   Debug.Print “High Accuracy Frequency = _
      ”;Format(Reading, “#.0000");” Hz"
End Sub
```

Example Output

```
Fast Frequency = 9998.2681 Hz
High Accuracy Frequency = 10000.0637 Hz
```


AP.Gen.FreqAccuracy

❶❷ Property

Syntax `AP.Gen.FreqAccuracy`

Data Type Integer

0	Set frequency accuracy mode to Fast. Fast mode produces the most rapid frequency settling along with frequency accuracy and resolution suitable for nearly all audio tests.
1	Set frequency accuracy mode to High Accuracy. High accuracy mode provides greater accuracy and resolution, but requires from 150 milliseconds (above 50Hz) to 750 milliseconds (at 10Hz) for complete settling each time the frequency is changed.

Description This command sets the frequency accuracy mode.

Fast mode produces the most rapid frequency settling along with frequency accuracy and resolution suitable for nearly all audio tests.

High accuracy mode provides greater accuracy and resolution, but requires from 150 milliseconds (above 50 Hz) to 750 milliseconds (at 10 Hz) for complete settling each time the frequency is changed.

Note that this command does not cause an immediate frequency calibration. The calibration will be done at the next call to `AP.Gen.Freq`.

See Also `AP.Gen.Freq`

Example See the example macro for `AP.Gen.Freq`.

AP.Gen.IMAmplRatio

❷ Property

Syntax **`AP.Gen.IMAmplRatio(`*`unit$`*`)`**

Data Type Double — Valid settings are 0.00001% to 100%

Parameters

Name	Description
`unit$`	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 between the High Frequency and the IM Frequency for the IMD (D/A) SMPTE waveform selection.

See Also `AP.Gen.IMHighFreq, AP.Gen.IMFreq`

AP.Gen.IMCenterFreq

❷ Property

Syntax **`AP.Gen.IMCenterFreq(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
`unit$`	String that designates the desired unit. The following unit is valid for this command. Hz

Description This command sets the Analog Generator IMD Center Frequency. The frequency passed is rounded to the closest available value.

Set the Analog Generator waveform to an IMD or IMD (D/A) CCIF before calling this command.

See Also AP.Gen.IMFreq

Example

```
Sub Main
   AP.Application.NewTest
   If AP.Application.SysType = 1 Then
      AP.Gen.Wfm(3)        'CCIF/DFD Waveform
      AP.Anlr.ChAInput = 1
   Else
      AP.Gen.Wfm(2,2)
      AP.Anlr.ChAInput = 2
   End If
   AP.Gen.IMCenterFreq("Hz") = 10000
   AP.Gen.IMFreq("Hz") = 80
   AP.Gen.Ampl("dBu") = 0.0
   AP.Gen.Output = 1
   AP.Anlr.FuncMode = 6    'CCIF measurement mode
   AP.Anlr.FuncInput = 0
   AP.Anlr.FuncSettling(3, .00003, "%", 3, .05, 1)
   AP.Anlr.FuncTrig
   Do
      Ready = AP.Anlr.FuncReady
      Loop Until Ready > 0
   Reading1 = AP.Anlr.FuncRdg("%")
   Debug.Print "CCIF/DFD = ";Format(Reading1, _
      "#.0000");" %"
End Sub
```


AP.Gen.IMFreq

❶❷ Property

Syntax **AP.Gen.IMFreq(*unit$*)**

Data Type Double

For a SMPTE mode waveform, this is the lower frequency tone. The following frequencies are available for System One:

500Hz, 250Hz, 125Hz, 100Hz, 60Hz, 50Hz, 40Hz
The following frequencies are available for System Two:
500Hz, 250Hz, 125Hz, 100Hz, 70Hz, 60Hz, 50Hz, 40Hz

For a CCIF mode waveform, this is the spacing between the two tones. The following frequencies are available: 1kHz, 500Hz, 250Hz, 200Hz, 120Hz, 100Hz, 80Hz

Parameters

Name	Description
unit$	String that designates the desired unit. The following unit is valid for this command. Hz

Description

This command sets the Analog Generator IMD frequency. The frequency passed is rounded to the closest available value.

Set the generator to an IMD waveform before calling this command in order to have the proper IMD frequency selected.

For a DIM mode waveform, this command has no effect. The frequencies are determined by the DIM mode selected. (See: `AP.Gen.Wfm`)

Because of frequency limitations, the actual frequency set may not be exactly what was requested. Therefore, when setting the IM Frequency it is important to check the returned frequency, and to use that value as the actual IM Frequency setting of the generator.

See Also

`AP.Gen.Wfm, AP.Gen.IMCenterFreq, AP.Gen.IMHighFreq`

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.Wfm(2,1)
   AP.Gen.IMHighFreq("Hz") = 7000
   AP.Gen.IMFreq("Hz") = 60
   AP.Gen.Ampl("dBu") = 0.0
   AP.Gen.Output = 1
   If AP.Application.SysType = "1" Then
      AP.Anlr.ChAInput = 1
   Else
      AP.Anlr.ChAInput = 2
   End If
   AP.Anlr.FuncMode = 5
   AP.Anlr.FuncInput = 0
   AP.Anlr.FuncSettling(3, .00003, "%", 3, .05, 1)
   AP.Anlr.FuncTrig
   Do
      Ready = AP.Anlr.FuncReady
```

```
      Loop Until Ready > 0
   Reading1 = AP.Anlr.FuncRdg("%")
   Debug.Print "SMPTE 4:1 = ";Format(Reading1,
"#.0000");" %"
End Sub
```

Example Output

```
SMPTE 4:1 = .0010 %
```

AP.Gen.IMHighFreq

❷ Property

Syntax **AP.Gen.IMHighFreq(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	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 or IMD (D/A) SMPTE before calling this command.

See Also AP.Gen.IMFreq

AP.Gen.Impedance

❶❷ Property

Syntax **AP.Gen.Impedance**

Data Type Integer

The following list contains the selections relevant to the AP.Gen.Config command for the Balanced and CMTST selections.

0	50 for System One, 40 for System Two
1	150
2	600

The following list contains the selections relevant to the `AP.Gen.Config` command for the Un-Balanced selections.

0	25 for System One, 20 for System Two
1	600

Description This command controls the output impedance for Balanced and Un-Balanced generator output configurations.

See Also `AP.Gen.Config`

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.Config = 0       'Set output configuration _
      to balanced floating.
   AP.Gen.Impedance = 2    'Set generator output _
      impedance to 600 ohms.
   AP.Gen.Ampl("dBm") = 0
   AP.Gen.Output = 1
   AP.Anlr.ChBRangeAuto = 0 'Set input ranging to fixed.
   AP.Anlr.ChBRange("V") = 2.5   'Set input range to _
      2.5 Volts.
   AP.Anlr.ChBInput = 0 'Set anlr input to INPUT(XLR).
   AP.Anlr.ChBImpedance = 1      'Set Cha B input Z to _
      600 ohms.
   AP.Anlr.FuncInput = 1   'Set Function Meter Cha to B.
   AP.Anlr.FuncSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.FuncTrig
   Do
      Ready = AP.Anlr.FuncReady
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg("dBm")
   Debug.Print "Channel B Amplitude = ";Format(Reading, _
      "#.0000");" dBm"
   AP.Anlr.ChBRangeAuto = 1   'Set input ranging to auto.
End Sub
```

Example Output

```
Channel B Amplitude = -105.9976 dBm
```

AP.Gen.Output ❶❷ Property

Syntax `AP.Gen.Output`

Data Type Boolean

True	On
False	Off

Description This command sets the Analog Generator channel A and B outputs to ON or OFF if they have been individually enabled by the `AP.Gen.ChAOutput` and `AP.Gen.ChBOutput` commands.

See Also `AP.Gen.ChAOutput, AP.Gen.ChBOutput`

Example See example for `AP.Gen.ChAAmpl`.

AP.Gen.Phase ❷ Property

Syntax **`AP.Gen.Phase(`***`unit$`***`)`**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: deg

Description This command sets the Analog Generator Phase value.

Set the Analog Generator waveform to Sine (D/A) Var Phase before calling this command.

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.Wfm 1, 1
   AP.Gen.Phase("deg") = 90.000000
   AP.Anlr.ChAInput = 2
   AP.Anlr.ChBInput = 2
   AP.Gen.Output = True
   Debug.Print "Channel B is " _
      & Format(AP.Anlr.PhaseRdg("deg"), "##.000") _
      & " deg relative to channel A."
```

```
End Sub
```

Example Output

```
Channel B is 89.984 deg relative to channel A.
```

AP.Gen.RefdBm

12 Property

Syntax **AP.Gen.RefdBm(*unit$*)**

Data Type Double — Impedance value.

Parameters

Name	Description
unit$	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 AP.Gen.Ampl command using the dBm unit, the software uses this dBm reference impedance value as the "R" in the *V2/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.*

See Also AP.Gen.Config, AP.Gen.Impedance

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.ChBOutput = 0    'Set generator output B to OFF.
   AP.Gen.Impedance = 1    'Set generator output Z to _
      150 Ohms.
   AP.Gen.RefdBm("Ohms") = 150   'Set dBm reference to _
      150 Ohms.
   AP.Gen.Ampl("dBm") = 0
   AP.Gen.Output = 1
   AP.Anlr.ChAImpedance = 0      'Set Cha A input Z to _
      150 ohms.
   AP.Anlr.RefdBm("Ohms") = 150  'Set dBm reference to _
      150 Ohms.
   Reference = AP.Anlr.RefdBm("Ohms")
   AP.Anlr.FuncSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.FuncTrig
   Do
```

```
      Ready = AP.Anlr.FuncReady
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg("dBm")
   Debug.Print "Channel A Amplitude = ";Format(Reading, _
      "#.0000");" dBm (";Reference;" Ohms)"
End Sub
```

Example Output

```
Channel A Amplitude = -94.2042 dBm ( 150 Ohms)
```

AP.Gen.RefdBr ❶❷ Property

Syntax **AP.Gen.RefdBr(*unit$*)**

Data Type Double — Amplitude value.

Parameters

Name	Description
unit$	V, dBu, dBV

Description This command sets the zero dBr value for the Analog Generator dBr units.

❶ Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.RefdBr("V") = 1
   AP.Gen.Ampl("dBr") = 1
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.ChBInput = 1
   AP.Anlr.RefdBr("V") = 1
   Reference = AP.Gen.RefdBr("V")
   AP.Anlr.ChALevelSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.ChALevelTrig
   Do
      Ready = AP.Anlr.ChALevelReady
   Loop Until (Ready > 0)
   Reading = AP.Anlr.ChALevelRdg("dBr")
   Debug.Print "Channel A Amplitude = ";Format(Reading, _
      "#.0000");" dBr relative to";Reference;" Volts"
End Sub
```

Example Output `Channel A Amplitude = .9679 dBr relative to 1 Volts`

AP.Gen.RefdBrAuto ❶❷ Method

Syntax **`AP.Gen.RefdBrAuto`**

Parameters None

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
   AP.Application.NewTest  'Reset panels
   AP.Gen.Ampl("dBV") = 0
   AP.Gen.RefdBrAuto
   AP.Gen.Ampl("dbr") = 2  'Increase amplitude 2 dB.
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.FuncMode = 0
   AP.Anlr.FuncInput = 0
   AP.Anlr.FuncSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.FuncTrig
   Do
      Ready = AP.Anlr.FuncReady
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg("dBV")
   Debug.Print "Channel A Amplitude ="; _
      Format$(Reading,"#.000000");" dBV"
End Sub
```

Example Output `Channel A Amplitude = 1.974047 dBV`

AP.Gen.RefFreq ❶❷ Property

Syntax **AP.Gen.RefFreq(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	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 AP.Gen.Freq

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = 1
   AP.Gen.RefFreq("Hz") = 5000
   AP.Gen.Freq("dHz") = 5000
   AP.Anlr.ChAInput = 1
   AP.Anlr.ChAFreqSettling(.5, .0002, "Hz", 3, .03, 1)
   AP.Anlr.ChAFreqTrig
   Do
      Ready = AP.Anlr.ChAFreqReady
   Loop Until Ready > 0
   Reading = AP.Anlr.ChAFreqRdg("Hz")
   Debug.Print "Frequency A = ";Format(Reading, _
      "#.0000");" Hz"
End Sub
```

Example Output

```
Frequency A = 9996.5878 Hz
```

AP.Gen.RefFreqAuto ❶❷ Method

Syntax **AP.Gen.RefFreqAuto**

Parameters None

Result Boolean

True	Frequenct 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
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = 1
   AP.Gen.RefFreqAuto
   AP.Gen.Freq("dHz") = 2000  'Increase frequency 2kHz.
   AP.Anlr.ChAInput = 1
   AP.Anlr.ChAFreqSettling(.5, .0002, "Hz", 3, .03, 1)
   AP.Anlr.ChAFreqTrig
   Do
      Ready = AP.Anlr.ChAFreqReady
   Loop Until Ready > 0
   Reading = AP.Anlr.ChAFreqRdg("Hz")
   Debug.Print "Frequency A ="; _
      Format$(Reading,"#.000000");" Hz"
End Sub
```

Example Output

```
Frequency A = 2998.543549 Hz
```

AP.Gen.RefWatts

02 Property

Syntax **AP.Gen.RefWatts(*unit$*)**

Data Type Double — Impedance value.

Parameters

Name	Description
unit$	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 AP.Gen.Ampl command using the Watts unit, the software uses this Watts reference impedance value as the "R" in the *VU!^X!O!A!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
   AP.Application.NewTest  'Reset panels
   AP.Gen.Output = 1
   AP.Gen.RefWatts("Ohms") = 8
   AP.Gen.Ampl("W") = .1
   AP.Anlr.ChAInput = 1
   AP.Anlr.RefWatts("Ohms") = 8
   AP.Anlr.FuncSettling(1, .000002, "V", 4, .05, 1)
   AP.Anlr.FuncTrig
   Do
      Ready = AP.Anlr.FuncReady
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg("W")
   Debug.Print "Output Power = ";Reading;" Watts"
End Sub
```

Example Output

```
Output Power =  9.92960921113392E-02 Watts
```


AP.Gen.Wfm

❶❷ Property

Syntax **AP.Gen.Wfm(***primary%, secondary%***)**

Data Type Integer

Parameters

Name	Description
primary%	This parameter defines the basic waveform type.
secondary%	This parameter defines the basic waveform modifier.

The following list is for System One.

Primary	Secondary	Description

0	Sine
1	SMPTE.DIN 1:1
2	SMPTE DIN 4:1
3	CCIF.DFD
4	DIM 30
5	DIM B
6	DIM 100
7	Square
8	Burst - Normal
9	Burst - Gated
10	Burst - Triggered
11	Noise - Pink
12	Noise - White
13	Noise - BP (Band Pass)
14	Pseudo Noise - Pink
15	Pseudo Noise - White
16	Pseudo Noise - BP (Band Pass)
17	DGEN
18	EQ Sine

The following list is for System Two.

Primary	Secondary	Description
0		Sine
	0	Normal
	1	Normal Burst
	2	Gated Burst
	3	Trig. Burst
	4	EQ Sine
1		Sine (D/A)
	0	Digital
	1	Var Phase
	2	Stereo
	3	Dual
	4	Shaped Burst
	5	EQ Sine

2		IMD
	0	SMPTE 1:1
	1	SMPTE 4:1
	2	CCIF/DFD
	3	DIM 30
	4	DIM B
	5	DIM 100
3		IMD (D/A)
	0	SMPTE/DIN
	1	CCIF/DFD
4		Square
5		Noise
	0	Pink - Pseduo
	1	White - Pseduo
	2	Pink BP - Pseduo
	3	Pink - Random
	4	White - Random
	5	Pink BP - Random
	6	Burst USASI
6		Arb Wfm (D/A)
7		MLS (D/A)
	0	Pink #1
	1	Pink #2
	2	Pink #3
	3	Pink #4
	4	White #1
	5	White #2
	6	White #3
	7	White #4
8		Special (D/A)
	0	Polarity

Description This command selects the Analog Generator waveform. The table above shows the possible settings for the `AP.Gen.Wfm` command.

Example

```
Sub Main
```

```
   AP.Application.NewTest  'Reset panels
   AP.Gen.Wfm(2,2)
   AP.Gen.Freq("Hz") = 13500
   AP.Gen.IMFreq("Hz")= 1000
   AP.Gen.Ampl("Vrms") = 2
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.FuncMode = 6
   AP.Anlr.FuncInput = 0
   AP.Anlr.FuncSettling(1, .00003, "%", 3, .05, 1)
   AP.Anlr.FuncTrig
   Do
      Ready = AP.Anlr.FuncReady
   Loop Until Ready > 0
   Reading = AP.Anlr.FuncRdg("dB")
   Debug.Print "CCIF = ";Format(Reading,"#.0000");" dB"
End Sub
```

Example Output

```
CCIF = -118.5611 dB
```

AP.Gen.WfmName

❷ Method

Syntax `AP.Gen.WfmName = filename$`

Data Type Boolean

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN waveform file (.agm or .ags).

Description This command loads the designated arbitrary waveform file (.AGM or .AGS) into the Digital Generator.

Note: This command can also be used to control the Digital Generator arbitrary waveform file selection.

See Also AP.Gen.Wfm

User Notes

User Notes

User Notes

User Notes

User Notes

Graph

AP.Graph.Comment ❶❷ Property

Syntax **AP.Graph.Comment**

Data Type String ASCII charactors.

Description This command transfers the ASCII charactors to or from the comment section in the Graph panel to a string variable.

See Also AP.Graph.CommentShow

❷ **Example**

```
Public Comment As String

Sub Main
   AP.Application.NewTest
   AP.File.OpenTest("COMMENT.AT2")
'Display Comment area In Graph
   AP.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)   'Display User Dialog
      Case 0
         'Remove Comment area from Graph
         AP.Graph.CommentShow = False
         End
      Case 1
         AP.Graph.Comment =  "Test Running"
         Wait .5
         AP.Sweep.Start
         Errors = AP.Data.LimitError(0)'Check for err
         If Errors >0 Then
            Comment = "Test FAILED"
```

```
            Else
               Comment = "Test PASSED"
            End If
      End Select
      GoTo DisplayDialog
End Sub

Private Function DlgHandler(DlgItem$, Action%,
SuppValue%) As Boolean
      Select Case Action%
      Case 1 ' Dialog box initialization
      Case 2 ' Value changing or button pressed
         DlgHandler = False 'Exit the dialog
      Case 3 ' TextBox or ComboBox text changed
      Case 4 ' Focus changed
      Case 5 ' Idle
         DlgHandler = True ' Continue getting idle actions
         AP.Graph.Comment = Comment$   'Display comment
         Wait .5
         AP.Graph.Comment = ""         'Remove comment
         Wait .2
      End Select
End Function
```

AP.Graph.CommentShow

❶❷ Property

Syntax **AP.Graph.CommentShow**

Data Type Boolean

True	Display Comment section.
False	Remove Comment section from view.

Description This command displayes or removes from view the comment section in the Graph panel

See Also AP.Graph.Comment

Example See example for AP.Graph.Comment.

AP.Graph.CopyToSweepPanel ❶❷ Method

Syntax **AP.Graph.CopyToSweepPanel**

Description 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.

❷ **Example**

```
Sub Main
   AP.Application.NewTest  'New Test
   AP.Gen.Output = True    'Generator Output ON
   AP.Anlr.ChAInput = 2 'Analyzer Ch A Input to GenMon
   AP.Anlr.ChBInput = 2 'Analyzer Ch B Input to GenMon
   AP.Anlr.FuncMode = 3    'Analyzer Function Meter _
      to THD+N Ampl

   AP.S2Dsp.Program = 2    'Select FFT Digital Analyzer
   AP.S2Dsp.FFT.InputFormat = 1'Select Low BW A/D Input
   AP.S2Dsp.FFT.Ch1Source = 2 'Digital Analyzer Ch 1 _
      Source to Anlr Rdg Ampl

   AP.Sweep.Data1.Id = 6024   'Select Fft.Ch.1 Ampl _
      for Data 1
   AP.Sweep.Data2.Id = 6027   'Select Fft.Ch.2 Ampl _
      for Data 2
   AP.Sweep.Source1.Id = 5515 'Select Fft.FFT Freq. _
      for Source 1
   AP.Sweep.Start             'Acqure waveform
   'Display data so that the vertical scaleing is _
      relative to optimized data for Data 1
   AP.Graph.OptimizeLeft      'Optimize Data 1
   AP.Graph.CopyToSweepPanel  'Copy Left and Right _
      graph vertical scale information to Sweep Panel
   AP.Sweep.CopyData1to2      'Copy Data 1 settings _
      to Data 2
   Wait 5
   'Display data so that the vertical scaleing is _
      relative to optimized data for Data 2
   AP.Graph.OptimizeRight     'Optimize Data 2
   AP.Graph.CopyToSweepPanel  'Copy Left and Right _
      graph vertical scale information to Sweep Panel
```

```
   AP.Sweep.CopyData2to1      'Copy Data 2 settings _
      to Data 1
   Wait 5
   'Display data so that the vertical scaleing is _
      optimized together for Data 1 and Data 2
   AP.Graph.OptimizeTogether  'Optimize Data 1 and _
      Data 2 Together
   Wait 5
   'Display data so that the vertical scaleing is _
      optimized individualy for Data 1 and Data 2
   AP.Graph.OptimizeIndividually 'Optimize Data 1 _
      and Data 2 Individually
End Sub
```

AP.Graph.CursorPosition ❶❷ Property

Syntax **AP.Graph.CursorPosition(***cursor%, unit$***)**

Data Type Double

Parameters

Name	Description
cursor%	1 = Cursor #1 2 = Cursor #2
unit$	Refer to the setting or reading defined by the *column* parameter to determine the appropriate unit selections.

Description This command returns the horizontal axis position value where the designated cursor is positioned.

See Also AP.Graph.CursorRow

❷ Example

```
Sub Main
   AP.Application.NewTest  'New Test
   AP.Gen.Output = True    'Generator Output ON
   AP.Anlr.ChAInput = 2 'Analyzer Ch A Input to GenMon
   AP.Anlr.ChBInput = 2 'Analyzer Ch B Input to GenMon

   AP.S2Dsp.Program = 2    'Select FFT Digital Analyzer
   AP.S2Dsp.FFT.InputFormat = 1'Select Low BW A/D Input
```

```
    AP.S2Dsp.FFT.Window = 5     'Select None, Move to _
       Bin Center

    AP.Sweep.Data1.Id = 6024    'Select Fft.Ch.1 Ampl _
       for Data 1
    AP.Sweep.Data2.Id = 6027    'Select Fft.Ch.2 Ampl _
       for Data 2
    AP.Sweep.Source1.Id = 5515 'Select Fft.FFT Freq. _
       for Source 1
    AP.Sweep.Start              'Acqure waveform

    AP.Graph.OptimizeTogether  'Optimize Data 1 and _
       Data 2 Together
    AP.Graph.CursorsOn(True)

    AP.Prompt.Text = "Select one of the Traces from _
       the Graph Legend then Position Cursor #1 on the _
       fundamental then press Continue."
    AP.Prompt.FontSize = 8      'Set font size to 8 point.
    AP.Prompt.Position(-1,-1,250,150) 'Set location and _
       size.
    AP.Prompt.ShowWithContinue 'Display prompt with _
       Continue button.
    Stop                        'Stop macro.

    Debug.Print "Frequency = " & _
       Format(AP.Graph.CursorPosition(1, "Hz"), _
       "##.0000")
    Debug.Print "Data Editor Row  = " & _
       AP.Graph.CursorRow(1)
    Debug.Print "Value = " &
       Format(AP.Graph.CursorValue(1, "V"), "##.0000")
End Sub
```

AP.Graph.CursorRow

❶❷ Property

Syntax **AP.Graph.CursorRow(***cursor%***)**

Data Type Integer

Parameters

Name	Description
cursor%	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 `AP.Data.Value` command.

See Also `AP.Graph.CursorPosition`

Example See example for `AP.Graph.CursorPosition`.

AP.Graph.CursorsOn ❶❷ Method

Syntax **`AP.Graph.CursorsOn`**

Data Type Boolean

True	Display cursors.
False	Remove cursors from view..

Parameters None

Result Boolean

True	Cursors displayed.
False	Cursors not displayed. This may be because the Graph Panel is not displayed.

Description This command displays or removes from view the cursors on the Graph panel.

Example See example for `AP.Graph.CursorPosition`.

AP.Graph.CursorValue ❶❷ Property

Syntax **`AP.Graph.CursorValue(`***`cursor%, unit$`***`)`**

Data Type Double

Parameters

Name	Description

cursor%	1 = Cursor #1 2 = Cursor #2
unit$	Refer to the setting or reading defined by the *column parameter to determine t e appropriate unit selections.*

Description This command returns the vertical axis value where the designated cursor is positioned.

See Also `AP.Graph.CursorPosition`

Example See example for `AP.Graph.CursorPosition`.

AP.Graph.OptimizeIndividually ❶❷ Method

Syntax **`AP.Graph.OptimizeIndividually`**

Parameters None

Result Void

Description This command optimizes the graph to display all data.

See Also `AP.Graph.OptimizeLeft, AP.Graph.OptimizeRight, AP.Graph.OptimizeTogether`

Example See example for `AP.Graph.CopyToSweepPanel`.

AP.Graph.OptimizeLeft ❶❷ Method

Syntax **`AP.Graph.OptimizeLeft`**

Parameters None

Result Void

Description This command optimizes the graph to display the data on the Left axis (Data 1).

See Also `AP.Graph.OptimizeIndividually, AP.Graph.OptimizeRight, AP.Graph.OptimizeTogether`

Example See example for `AP.Graph.CopyToSweepPanel`.

AP.Graph.OptimizeRight ❶❷ Method

Syntax `AP.Graph.OptimizeRight`

Parameters None

Result Void

Description This command optimizes the graph to display the data on the Right axis (Data 2).

See Also `AP.Graph.OptimizeIndividually, AP.Graph.OptimizeLeft, AP.Graph.OptimizeTogether`

Example See example for `AP.Graph.CopyToSweepPanel`.

AP.Graph.OptimizeTogether ❶❷ Method

Syntax `AP.Graph.OptimizeTogether`

Parameters None

Result Void

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 `AP.Graph.OptimizeIndividually, AP.Graph.OptimizeLeft, AP.Graph.OptimizeRight`

Example See example for `AP.Graph.CopyToSweepPanel`.

AP.Graph.RefDataClear ❶❷ Method

Syntax `AP.Graph.RefDataClear`

Parameters None

Result Void

Description This command Clears all reference data from memory. Clearing nonexistent reference data does not produce an error.

See Also `AP.Graph.RefDataShow, AP.Graph.RefDataStore`

Example

AP.Graph.RefDataShow ❶❷ Property

Syntax **`AP.Graph.RefDataShow`**

Data Type Boolean

`True`	Display Reference Data.
`False`	Remove Reference Data from view..

Description This command makes the Reference Data visable or invisible on the graph

See Also `AP.Graph.RefDataClear, AP.Graph.RefDataStore`

Example See example for `AP.Graph.RefDataClear`.

AP.Graph.RefDataStore ❶❷ Method

Syntax **`AP.Graph.RefDataStore`**

Parameters None

Result Void

Description This command adds the Sweep Data currently in memory to Reference Data memory.

See Also `AP.Graph.RefDataClear, AP.Graph.RefDataShow`

Example See example for `AP.Graph.RefDataClear`.

User Notes

User Notes

User Notes

User Notes

Log

AP.Log.AddEntry ❶❷ Method

Syntax **AP.Log.AddEntry(*string$*)**

Parameters

Name	Discription
string$	Any valid string

Result Void

Description This command appends the defined string to the log file.

❶ **Example**

```
Sub Main
   AP.Log.Enable = True          'Enable logging.
   'Set log file to "S1-22CK.ALG"
   AP.Log.FileName = "S1-22CK.ALG"
   AP.Log.ErrorMessages = True   'Log error messages.
   AP.Log.FileActivity = True    'Log File Input/Out.
   AP.Log.PassFailMessages = True'Log Pass/Fail _
      messages.
   AP.Log.TestName = True        'Log test name.
   AP.Log.GraphTitle = True      'Log graph title.
   AP.Log.Data = True            'Log all data.
   AP.Log.Clear                  'Clear log file.
   AP.Log.AddEntry "Amplitude Linearity."
   AP.File.OpenTest "S1AMPLIN.AT1" 'Open test.
   AP.Sweep.Start                'Start sweep.
   AP.Log.View                   'View log file.
End Sub
```

Example Output

```
     01/09/96 14:27:01  Amplitude Linearity.

     01/09/96 14:27:01  Open Test: S1AMPLIN.AT1
PASS 01/09/96 14:27:02  Execute sweep: S1AMPLIN.AT1
```

Comment The example output is from the log file created when the example macro is run.

AP.Log.Clear

❶❷ Method

Syntax `AP.Log.Clear`

Parameters None

Result Void

Description This command clears the contents of the current log file.

Example See example for `AP.Log.AddEntry`.

AP.Log.Data

❶❷ Property

Syntax `AP.Log.Data`

Data Type Integer

0	None
1	All
2	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 `AP.Log.AddEntry`.

AP.Log.Enable

❶❷ Property

Syntax `AP.Log.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 `AP.Log.AddEntry`.

AP.Log.ErrorMessages ❶❷ Property

Syntax **`AP.Log.ErrorMessages`**

Data Type Boolean

`True`	Enable
`False`	Disable

Description This command when enabled logs into the log file any APWIN or windows error messages which occur durring the period that logging is enabled.

Example See example for `AP.Log.AddEntry`.

AP.Log.FileActivity ❶❷ Property

Syntax **`AP.Log.FileActivity`**

Data Type Boolean

`True`	Enable
`False`	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 `AP.Log.AddEntry`.

AP.Log.FileName ❶❷ Property

Syntax **`AP.Log.FileName(`***`filename$`***`)`**

Parameters

Name	Discription
filename$	Any valid DOS filename and extension.

Description This command defines the file name to use for the log file.

Example See example for `AP.Log.AddEntry`.

AP.Log.GraphTitle

❶❷ Property

Syntax `AP.Log.GraphTitle`

Data Type Boolean

True	Enable
False	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 `AP.Log.AddEntry`.

AP.Log.PassFailMessages

❶❷ Property

Syntax `AP.Log.PassFailMessages`

Data Type Boolean

True	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 `AP.Log.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 `AP.Log.AddEntry`.

AP.Log.PrintLogFile ❶❷ Method

Syntax **`AP.Log.PrintLogFile`**

Parameters None

Result Void

Description This command loads the NOTEPAD application and prints the current log file and then closes the NOTEPAD application.

❶ **Example**

```
Sub Main
   AP.Log.Enable = True          'Enable logging.
   AP.Log.FileName = "S1-22CK.ALG" 'Set log file to _
      "S1-22CK.ALG"
   AP.Log.ErrorMessages = True   'Log error messages.
   AP.Log.FileActivity = True    'Log File Input/Out.
   AP.Log.PassFailMessages = True'Log Pass/Fail messages.
   AP.Log.TestName = True        'Log test name.
   AP.Log.GraphTitle = True      'Log graph title.
   AP.Log.Data = True            'Log all data.
   AP.Log.Clear                  'Clear log file.
   AP.Log.AddEntry "Amplitude Linearity."
   AP.File.OpenTest "S1AMPLIN.AT1" 'Open test.
   AP.Sweep.Start                'Start sweep.
   AP.Log.PrintLogFile           'Print log file.
End Sub
```

AP.Log.TestName ❶❷ Property

Syntax **`AP.Log.TestName`**

Data Type Boolean

True	Enable
False	Disable

Description This command when enabled logs the test name, including path name, of the test when executed.

Example See example for `AP.Log.AddEntry`.

AP.Log.View ❶❷ Method

Syntax `AP.Log.View`

Parameters None

Result Void

Description This command loads the NOTEPAD application and displays the current log file.

Example See example for `AP.Log.AddEntry`.

User Notes

User Notes

Macro

AP.Macro.IsRunning OLE ❶❷ Method

Syntax **AP.Macro.IsRunning**

Parameters None

Result Boolean

True	APWIN Macro running.
False	APWIN Macro not running.

Description This command returns the status of the APWIN macro.

Example

```
Private Sub Form_Load()
   Dim AP As Object
   'Create OLE link to APWIN.
   Set AP = CreateObject("APWIN.Application")
   AP.Application.Visible = True ' Make APWIN visable

   'Place your code here

   'Run an APWIN Macro and wait for it to finish
   AP.File.OpenMacro "C:\BUSY.APB"
   While AP.Macro.IsRunning = True
   Wend

   'Change Visual Basic directory to APWIN Working _
      Directory.
   ChDir AP.Application.MacroDir

   'Place your code here

   AP.Application.Quit           'Quit APWIN
   End
End Sub
```

AP.Macro.Name

❶❷ Method

Syntax **AP.Macro.Name**

Result Boolean ASCII charactors.

Description This command returns the APWIN Macro Editor Name. This text string is located in the upper left corner of the APWIN Macro/Procedure Editor before the Macro/Procedure name. This string is usefull when using the AppActivate command in the Language reference section of APWIN Basic.

See Also AP.Application.Name

Example

```
Sub Main
   AppActivate AP.Application.Name  'Select the APWIN _
      window
   'The following SendKey command will now be sent to _
      the APWIN application.
   SendKeys "%WC",1  'Clear all windows on page.
   SendKeys  "%PO",1 'Display Data Editor.
   AppActivate AP.Macro.Name  'Change focus back to _
      the Procedure/Macro 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
```

User Notes

User Notes

User Notes

User Notes

Print

AP.Print.Data

1 2 Method

Syntax **AP.Print.Data**

Parameters None

Result Void

Description This command prints the data displayed in the Data Editor in tabular format. The Data Editor must be displayed on at least one of the APWIN Pages. The printer is defined by the settings on the File, Print Setup menu.

See Also AP.Print.Graph

Example

```
Sub Main
   System = AP.Application.SysType
   If System = "1" Then
   'Load test containing measurement data
      AP.File.OpenTest("GRAPH.AT1")
   Else
   'Load test containing measurement data
      AP.File.OpenTest("GRAPH.AT2")
   End If
   AP.Application.PanelOpen apbPanelDataEditor
   AP.Sweep.Start
   AP.Print.Data  'Print Data in tabular form
End Sub
```

AP.Print.Graph

1 2 Method

Syntax **AP.Print.Graph**

Parameters None

Result Void

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 APWIN Pages to print or preview the graph.

See Also `AP.Print.LoadFromTest`

Example

```
Sub Main
   System = AP.Application.SysType

   If System = “1" Then
      ‘Load test containing graph setup
      AP.File.OpenTest(“GRAPH.AT1")
      AP.Print.LoadFromTest
      AP.File.OpenTest(“TEST.AT1")
   Else
      ‘Load test containing graph setup
      AP.File.OpenTest(“GRAPH.AT2")
      AP.Print.LoadFromTest
      AP.File.OpenTest(“TEST.AT2")
   End If

   AP.Sweep.Start
   AP.Print.Graph
End Sub
```

AP.Print.LoadFromTest ❶❷ Method

Syntax `AP.Print.LoadFromTest`

Parameters None

Result Void

Description This command loads the page setup settings from the currrently 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 consistant format over many different tests. Loading a test doesn’t

automaticly update the Page Setup menu and change the graphic output.

See Also `AP.Print.Graph`

Example See example macro `AP.Print.Graph`.

AP.Print.TrackGraph

❶❷ Method

Syntax **`AP.Print.TrackGraph`**

Parameters None

Result Void

Description 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 APWIN Pages to print or preview the graph.

See Also `AP.Print.LoadFromTest`

Example

```
Sub Main
   AP.Application.NewTest      'Start with New Test
   AP.Print.TrackGraph = True 'Have graph printout _
      track graph panel legend settings
   AP.Gen.Output = True        'Turn On Analog Generator
   AP.Anlr.ChAInput = 2        'Have Analog Analyzer _
      monitor Generator Channel A
   AP.Anlr.FuncFilterHP = 3    'Select 400Hz High Pass _
      Filter
   AP.Sweep.Start              'Run Sweep
   AP.Print.Graph              'Print graph
End Sub
```

User Notes

User Notes

User Notes

Prompt

AP.Prompt.FontSize ❶❷ Property

Syntax `AP.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
   'Set string to display in prompt.
   AP.Prompt.Text = "This prompt will be shown for 5 _
      seconds."
   AP.Prompt.FontSize = 8     'Set font size to 8 point.
   AP.Prompt.Position(-1,-1,190,120)'Set location and _
      size.
   AP.Prompt.Show             'Show prompt.
   Shown = AP.Prompt.IsUp
   Wait 5                     'Wait 5 seconds.
   AP.Prompt.Hide             'Hide prompt.
   Shown = AP.Prompt.IsUp
   AP.Prompt.Text = "This prompt will be shown until _
      the Continue Macro button is selected _
      below."  'Set string to display in prompt.
   AP.Prompt.ShowWithContinue 'Display prompt with _
      Continue button.
   Stop                       'Stop macro.
End Sub
```

AP.Prompt.Hide ❶❷ Method

Syntax `AP.Prompt.Hide`

Parameters None

Result Void

Description This command removes the user prompt from view.

Example See example macro `AP.Prompt.FontSize`.

AP.Prompt.IsUp

❶❷ Method

Syntax **`AP.Prompt.IsUp`**

Parameters None

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 `AP.Prompt.FontSize`.

AP.Prompt.Position

❶❷ Method

Syntax **`AP.Prompt.Position(`***x%*, *y%*, *dx%*, *dy%*`)`

Parameters

Name	Description
x%	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.
y%	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.
dx%	This number value is the width. It is measured in 1/8ths of the average character width for the dialog's font.
dy%	This number value is the height. It is measured in 1/12ths of the character height for the dialog's font.

Result Integer

Description This command defines the position and size of the User Prompt.

Example See example for `AP.Prompt.FontSize`.

AP.Prompt.Show

❶❷ Method

Syntax `AP.Prompt.Show`

Parameters None

Result Void

Description This command displays the user prompt.

See Also `(Language Reference) Stop Instruction`

Example See example for `AP.Prompt.FontSize`.

AP.Prompt.ShowWithContinue

❶❷ Method

Syntax `AP.Prompt.ShowWithContinue`

Parameters None

Result Void

Description This command displays the current user prompt window with a continue button.

See Also `(Language Reference) Stop Instruction`

Example See example for `AP.Prompt.FontSize`.

AP.Prompt.ShowWithContinueAndStopSweep

❶❷ Method

Syntax `AP.Prompt.ShowWithContinueAndStopSweep`

Parameters None

Result Void

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
   AP.Application.NewTest  'Reset panels
   AP.Gen.Ampl("Vrms") = 2
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.Anlr.FuncMode = 0
   AP.Anlr.FuncInput = 0

   AP.Prompt.Text = "Press the Continue button to STOP _
      the SWEEP and continue the Macro."
   AP.Prompt.FontSize = 8  'Set font size to 8 point.
   AP.Prompt.Position (-1,-1,180,120)  'Set location and _
      size.
   'Display prompt with Continue and also stop the _
      current sweep.
   AP.Prompt.ShowWithContinueAndStopSweep
   AP.Sweep.Start                'Start Sweep.

   AP.Prompt.Text = "Press the Continue button to END _
      the Macro."
   AP.Prompt.FontSize = 8     'Set font size to 8 point.
   AP.Prompt.Position (-1,-1,200,100)  'Set location and _
      size.
   AP.Prompt.ShowWithContinue       'Display prompt with _
      Continue.
   Stop
End Sub
```

AP.Prompt.Text

02 Property

Syntax AP.Prompt.Text

Data Type String User defined string.

Description This command defines the string to be displayed in a user prompt.

Example See example for `AP.Prompt.FontSize`.

User Notes

User Notes

User Notes

Regulation

AP.Reg.IsRunning

❶❷ Method

Syntax `AP.Reg.IsRunning`

Parameters None

Result Boolean

True	Regulation process running.
False	Regulation process not running.

Description This command returns the status of the Regulation process.

See Also `AP.Macro.LoadFromTest`

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.ChAAmpl("dBV") = 0.0
   AP.Application.PanelOpen apbPanelRegulation
   'Regulate Analyzer Level A to -80 dBV to within a _
      Tolerance 1.0dB
   AP.Reg.TargetID = 5903  'Regulate Analyzer Level A _
      to -80 dBV to within a Tolerance 1.0dB
   AP.Reg.TargetValue("dBV") = -80.000000
   AP.Reg.TargetToleranceMode = 1   'Tolerance Mode dB
   AP.Reg.TargetTolerance("dB") = 1.0 'Tolerance 1.0dB

   AP.Reg.SourceID = 5052  'by varying the Generator _
      A Amplitude
   AP.Reg.SourceHigh("dBV") = -70.0 'High Amplitude _
      boundary
   AP.Reg.SourceLow("dBV") = -90.0  'Low Amplitude _
      boundary
   AP.Reg.SourceOperation = 2       '-Normal Operation
   AP.Reg.SourceStepSize("dB") = 0.1
   AP.Reg.SourceIteration = 30

   AP.Reg.StartNoWait(True)'Start the Regulation _
      process then continue.
```

```
   StartTime = Timer
   Do    'Wait until Regulation process is finished _
      or Timeout has elapsed.
      Wait .1
      Debug.Print Timer
   Loop While AP.Reg.IsRunning And Timer < StartTime _
      + AP.Reg.Timeout
   If AP.Reg.IsRunning = True Then  'Stop Regulation _
      process if running
      AP.Reg.StartNoWait(False)
      Debug.Print "Regulation Stopped!"
   End If
End Sub
```


AP.Reg.SourceHigh

❶❷ Property

Syntax **AP.Reg.SourceHigh(*unit$*)**

Data Type Double — Refer to the setting defined by the AP.Reg.SourceId command (ID#) to determine the appropriate range of acceptable values.

Parameters

Name	Description
unit$	Refer to the setting defined by the AP.RegSourceId conmmand (ID#) to determine the appropriate unit selections.

Description This command sets the upper boundary for the source parameter used in the regulation process.

See Also AP.Reg.SourceLow

Example See example for AP.Reg.SourceId.

AP.Reg.SourceId

❶❷ Property

Syntax **AP.Reg.SourceId**

Data Type Long

Instrument Parameter ID#.

Description This command is used to select the instrument parameter which will define settings used in the regulation process.

Refer to Appendix D to obtain instrument parameter identification numbers.

❷ **Example**

```
Sub Main
   AP.File.OpenTest  "Reg3.at2"
   AP.Reg.SourceID = 5051      'Set Source To GenFreq
   AP.Reg.TargetID = 5906      'Set Source to AnlrAmpl
   AP.Reg.SourceOperation = 1 'Set Operation to +Normal
   AP.Reg.SourceStepSize("%") = 2 'Set Stepsize to 2%
   AP.Reg.SourceIteration = 30    'Set iterations to 30
   AP.Reg.TargetToleranceMode = 0   'Tolerance Mode to %
   AP.Reg.TargetValue("dBrA") = -3  'Tolerance to -3
   AP.Reg.TargetTolerance("%") = 5  'Tolerance units %
   AP.Reg.SourceHigh("Hz") = 5000   'High Bound to 5kHz
   AP.Reg.SourceLow("Hz") = 200     'Low Bound to 20 Hz
   AP.Reg.SweepEnable = False    'Don't Regulate each _
      step in sweep.
   AP.Reg.Timeout = 2.0          'Terminate regulation _
      process if timed out
   AP.Reg.Start                  'Start Regulation
End Sub
```


AP.Reg.SourceIteration ❶❷ Property

Syntax `AP.Reg.SourceIteration`

Data Type Long

Description 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.

See Also `AP.Reg.SourceOperation`

Example See example for `AP.Reg.SourceId`.

AP.Reg.SourceLow

❶❷ Property

Syntax `AP.Reg.SourceLow(unit$)`

Data Type Double

Refer to the setting defined by the `AP.Reg.SourceId` command (ID#) to determine the appropriate range of acceptable values.

Parameters

Name	Description
unit$	Refer to the setting defined by the `AP.RegSourceId` conmmand (ID#) to determine the appropriate unit selections.

Description This command sets the lower boundary for the source parameter used in the regulation process.

See Also `AP.Reg.SourceHigh`

Example See example for `AP.Reg.SourceId`.

AP.Reg.SourceOperation

❶❷ Property

Syntax `AP.Reg.SourceOperation`

Data Type Integer

0	Linear: assumes a linear relationship between the source setting and the target reading.
1	+ Normal: assumes that an increasing source setting will cause an increasing target reading, but not necessarily linearly.
2	- Normal: assumes that an increasing source setting will cause a decreasing target reading, but not necessarily linearly.
3	Maximum: 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 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 untill 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 `AP.Reg.SourceIterations` command.
4	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 `AP.Reg.SourceIterations` command.

Description This command selects the type of algorithm used to control the source parameter specified by the `AP.Reg.SourceId` command.

See Also `AP.Reg.SourceStepSize, AP.Reg.SourceIteration`

Example See example for `AP.Reg.SourceId`.

AP.Reg.SourceStepSize ❶❷ Property

Syntax **`AP.Reg.SourceStepSize(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
`unit`	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 `AP.Reg.SourceOperation, AP.Reg.SourceIteration`

Example See example for `AP.Reg.SourceId`.

AP.Reg.Start

❶❷ Method

Syntax `AP.Reg.Start`

Description This command initiates or terminates a regulation process.

Example See example for `AP.Reg.SourceId`.

AP.Reg.StartNoWait

❶❷ Method

Syntax `AP.Reg.StartNoWait`

Result Boolean

True	Start regulation process.
False	Stop regulation process.

Description This command initiates a regulation process and then continues macro execution.

Example See example for `AP.Reg.SourceId`.

AP.Reg.SweepEnable

❶❷ Property

Syntax `AP.Reg.SweepEnable`

Data Type Boolean

True	Enable regulation for each sweep step.
False	Disable regulation for each sweep step.

Description This command enables or disables regulation before each step in a sweep.

See Also `AP.Reg.Start`

Example See example for `AP.Reg.SourceId`.

AP.Reg.TargetId ❶❷ Property

Syntax `AP.Reg.TargetId`

Data Type Long — Instrument Parameter ID#.

Description This command is used to select the instrument parameter which will return readings used in the regulation process.

Refer to Appendix D to obtain instrument parameter identification numbers.

Example See example for `AP.Reg.SourceId`.

AP.Reg.TargetTolerance ❶❷ Property

Syntax `AP.Reg.TargetTolerance(unit$)`

Data Type Double — Refer to the setting defined by the `AP.Reg.TargetId` command (ID#) to determine the appropriate range of acceptable values.

Parameters

Name	Description
unit$	Refer to the setting defined by the `AP.RegTargetId` conmmand (ID#) to determine the appropriate unit selections for the Abs selection for the `AP.Reg.TargetToleranceMode` command. The following units are avalible when % is selected with the `AP.Reg.TargetToleranceMode` command: X/Y, %

Description This command sets the tolerance that the regulation algorithm uses to determine if the regulation process is complete.

See Also `AP.Reg.TargetToleranceMode, AP.Reg.TargetValue`

Example See example for `AP.Reg.SourceId`.

AP.Reg.TargetToleranceMode

❶❷ Property

Syntax `AP.Reg.TargetToleranceMode`

Data Type Integer

0	%
1	dB
2	Abs

Description This command selects the type of units the regulation algorithm uses to specify the Target Tolerance.

See Also `AP.Reg.TargetTolerance, AP.Reg.TargetValue`

Example See example for `AP.Reg.SourceId`.

AP.Reg.TargetValue

❶❷ Property

Syntax `AP.Reg.TargetValue(unit$)`

Data Type Double — Refer to the setting defined by the `AP.Reg.TargetId` command (ID#) to determine the appropriate range of acceptable values.

Parameters

Name	Description
unit$	Refer to the setting defined by the `AP.RegTargetId` conmmand (ID#) to determine the appropriate unit selections.

Description This command sets the Target value that the regulation algorithm attempts to obtain.

See Also `AP.Reg.TargetTolerance, AP.Reg.TargetToleranceMode`

Example See example for `AP.Reg.SourceId`.

AP.Reg.TimeOut

❶❷ Property

Syntax `AP.Reg.Timeout`

Data Type Double

Description This command sets the period of time allowed to complete each regulation process.

Example See example for `AP.Reg.SourceId`.

User Notes

User Notes

User Notes

User Notes

User Notes

APWIN BASIC
User's Manual and Programmer's Reference

Volume 1 Language Reference

Volume 2 Extensions A-R

Volume 3 Extensions S-Z

Version 1.52
November, 1998

Version 1.52 November, 1998

Published by:

Audio precision

Audio Precision, Inc.
PO Box 2209
Beaverton, Oregon 97075-2209
U.S. Toll Free: 1-800-231-7350
Tel: (503) 627-0832 Fax: (503) 641-8906
email: techsupport@audioprecision.com
Web: www.audioprecision.com

Printed in the United States of America
Audio Precision Part # 8211-0025

System One Digital Input/Output

AP.S1Dio.DitherType

Property

Syntax `AP.S1Dio.DitherType`

Data Type Integer

0	Triangular: Triangular probability function dither has no noise modulation effect but produces a slightly worse output signal to noise ratio. This is normally the preferred choice.
1	Rectangular: Rectangular probability function dither provides the best signal to noise, but suffers from modulation noise effects.
2	Shaped: Shaped dither 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.
3	None

Description This command sets the type of dither to be applied to the selected digital output.

The dither amplitude is automatically determined by the value entered into the Resolution field. Rectangular probability dither amplitude is set at 1/2 LSB and triangular dither amplitude is set at one LSB.

Example

```
Sub Main
   System = AP.Application.SysType
   If System = "2" Then Debug.Print"System One Example _
      only."
   AP.Application.NewTest
   AP.S1Dsp.Program = 2
   AP.S1Dio.InFormat = 2
   AP.S1Dio.OutFormat = 1
   AP.S1Dio.SerialType = 0
   AP.S1Dio.Rate = 3
   AP.S1Dio.DitherType = 3
   AP.S1Dio.Resolution = 24
```

```
   If AP.S1Dio.TransmitSyncRdg Then
      Debug.Print "The digital output sample rate is _
         locked to the sync input frequency."
   Else
      Debug.Print "The digital output sample rate is _
         NOT locked to the sync input frequency."
   End If
   If AP.S1Dio.ReceiveSyncRdg Then
      Debug.Print "An acceptable digital signal is _
         connected to the input connector."
   Else
      Debug.Print "An acceptable digital signal is _
         NOT connected to the input connector."
   End If
End Sub
```

Example Output

```
The digital output sample rate is locked to the sync _
   input frequency.
An acceptable digital signal is connected to the _
   input connector.
```

AP.S1Dio.InFormat

● Property

Syntax `AP.S1Dio.InFormat`

Data Type Integer

0	Serial
1	Parallel
2	DGEN

Description This command selects the digital input format.

See Also `AP.S1Dio.OutFormat`

Example See example for `AP.S1Dio.DitherType`.

AP.S1Dio.OutFormat

Property

Syntax `AP.S1Dio.OutFormat`

Data Type Integer

0	D/A
1	Serial
2	Parallel

Description This command selects the digital output format.

See Also `AP.S1Dio.InFormat`

Example See example for `AP.S1Dio.DitherType`.

AP.S1Dio.Rate

Property

Syntax `AP.S1Dio.Rate`

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.Program` command Digital Domain Audio Tester (genanlr), Triggered Multitone Tester (fasttrig), Quasi-anechoic Acoustical Tester (mls), Digital Data Analyzer (bittest), and Codec Tester (codec) selections.

0	32k
1	48k
2	44.1k

The following list contains the selections relevant to the `AP.S1DSP.Program` command Spectrum Analyzer/Generator (fftgen), and Spectrum Analyzer (fftslide) selections.

0	1k
1	8k
2	32k
3	48k

4	192k
5	44.1k
6	176k

The following list contains the selections relevant to the `AP.S1DSP.Program` command Multitone Generator/ Analyzer (fasttest) selection.

0	8k
1	32k
2	48k
3	44.1k

The following list contains the selections relevant to the `AP.S1DSP.Program` command Narrow Bandpass Filter (harmonic) selection.

0	48k
1	192k

Description This command selects the Digital Input/Output Sample Rate for the System One DSP programs.

See Also `AP.S1DSP.Program`

Example See example for `AP.S1Dio.DitherType`.

AP.S1Dio.ReceiveSyncRdg

❶ Method

Syntax **`AP.S1Dio.ReceiveSyncRdg`**

Parameters None

Result Boolean

True	Indicates that an acceptable digital signal is connected to the input connector.
False	Indicates that an acceptable digital signal is not connected to the input connector.

Description This command returns the status of the Receive Sync indicator located on the System One Digital IO Parameters Panel.

The Receive Sync indicator will light if an acceptable digital signal is connected to the input connector selected in the Input Format field.

See Also `AP.S1Dio.TransmitSyncRdg`

Example See example for `AP.S1Dio.DitherType`.

AP.S1Dio.Resolution

Property

Syntax **`AP.S1Dio.Resolution`**

Data Type Integer

Parameters None

Description This command determines the output word width. Digital output signals are internally generated at a full 24 bit resolution in System One, then rounded to the bit value specified in this field. If dither is turned on, the dither will also be added at the LSB (least significant bit) level of the specified resolution.

Example See example for `AP.S1Dio.DitherType`.

AP.S1Dio.SerialType

Property

Syntax **`AP.S1Dio.SerialType`**

Data Type Integer

0	AES/EBU & SPDIF
1	GP Serial

Description This command selects the Serial interface type.

See Also `AP.S1Dio.InFormat, AP.S1Dio.OutFormat`

Example See example for `AP.S1Dio.DitherType`.

AP.S1Dio.TransmitSyncRdg

Method

Syntax `AP.S1Dio.TransmitSyncRdg`

Parameters None

Result Boolean

True	Indicates that the digital output sample rate is locked to the sync input frequency.
False	Indicates that the digital output sample rate is not locked to the sync input frequency.

Description This command returns the status of the Transmit Sync indicator located on the System One Digital IO Parameters Panel.

The Transmit Sync indicator shows whether an acceptable signal has been connected to the sync input on the rear of the chassis. When this command returns True, the digital output sample rate is locked to the sync input frequency rather than being controlled by the Rate field (False) or `AP.S1Dio.Rate` command.

See Also `AP.S1Dio.ReceiveSyncRdg`

Example See example for `AP.S1Dio.DitherType`.

User Notes

User Notes

User Notes

Digital Data Analyzer

AP.S1DSP.Bittest.Ampl

Property

Syntax **AP.S1DSP.Bittest.Ampl(*unit$*)**

Data Type Double — Valid amplitude settings are from 0 to 100 %FS.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS

Description This command sets the Digital Data Analyzer Digital Generator Amplitude.

Example

```
Sub Main
   AP.File.OpenTest "BITTEST1.AT1"     'Open test
   AP.S1DSP.Bittest.Output = True
   AP.S1DSP.Bittest.Freq("Hz") = 1000
   AP.S1DSP.Bittest.Ampl("FFS") = .995
   AP.S1DSP.Bittest.ChAOutput = True
   Wait 1
   AP.S1DSP.Bittest.ChADataTrig  'Trigger a new reading
   Do
      Ready = AP.S1DSP.Bittest.ChADataReady
   Loop Until Ready > 0       'Wait for new reading
   Reading1 = AP.S1DSP.Bittest.ChADataRdg("dec") 'Get _
      new reading
   NewLine$ = Chr(13)
   a$= "Ch 1 Data "+Left(Str$(Reading1),6)+"dec"
   AP.Prompt.Text = a$ + NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.Bittest.ChADataRdg

Property

Syntax `AP.S1DSP.Bittest.ChADataRdg(unit$)`

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: dec.

Description This command returns a unsettled reading for the Digital Data Analyzer channel A Data meter and zeros the ready count.

See Also `AP.S1DSP.Bittest.ChADataReady`, `AP.S1DSP.Bittest.ChADataTrig`

Example

Example Output See example for `AP.S1DSP.Bittest.Ampl`.

AP.S1DSP.Bittest.ChADataReady

Property

Syntax `AP.S1DSP.Bittest.ChADataReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Data Analyzer channel A Data 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 `AP.S1DSP.Bittest.ChADataRdg` or `AP.S1DSP.Bittest.ChADataTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.Bittest.ChADataRdg` command will be guaranteed to return quickly.

See Also AP.S1DSP.Bittest.ChADataRd, AP.S1DSP.Bittest.ChADataTrig

Example Output See example for AP.S1DSP.Bittest.Ampl.

AP.S1DSP.Bittest.ChADataTrig

Method

Syntax **AP.S1DSP.Bittest.ChADataTrig**

Description Causes a restart of the reading cycle and zeros the ready count for the AP.S1DSP.Bittest.ChADataRdg command. The reading in progress is aborted.

See Also AP.S1DSP.Bittest.ChADataRdg, AP.S1DSP.Bittest.ChADataReady

Example Output See example for AP.S1DSP.Bittest.Ampl.

AP.S1DSP.Bittest.ChAErrRdg

Property

Syntax **AP.S1DSP.Bittest.ChAErrRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: dec.

Description This command returns a unsettled reading for the Digital Data Analyzer channel A Errors meter and zeros the ready count.

See Also AP.S1DSP.Bittest.ChAErrReady, AP.S1DSP.Bittest.ChAErrTrig

Example

```
Sub Main
   AP.File.OpenTest "BITTEST1.AT1"          'Open test
   AP.S1DSP.Bittest.Output = True
   AP.S1DSP.Bittest.Ampl("FFS") = .995
   AP.S1DSP.Bittest.ChAOutput = True
   AP.S1DSP.Bittest.ChBOutput = True
```

```
   AP.S1DSP.Bittest.DisplayError = 0   'Set Error _
      Display Normal
   AP.S1DSP.Bittest.RdgRate = 0  'Set Reading Rate _
      to Auto
   AP.S1DSP.Bittest.ChAErrTrig   'Trigger a new reading
   Do
      Ready = AP.S1DSP.Bittest.ChAErrReady
   Loop Until Ready > 0          'Wait for new reading
   Reading1 = AP.S1DSP.Bittest.ChAErrRdg("dec") 'Get _
      new reading
   NewLine$ = Chr(13)
   a$= "Ch 1 Errors "+Left(Str$(Reading1),6)+"dec"
   AP.Prompt.Text = a$ + NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.Bittest.ChAErrReady

Property

Syntax **AP.S1DSP.Bittest.ChAErrReady**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Data Analyzer channel A Errors 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 `AP.S1DSP.Bittest.ChAErrRdg` or `AP.S1DSP.Bittest.ChAErrTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.Bittest.ChAErrRdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.Bittest.ChAErrRdg`, `AP.S1DSP.Bittest.ChAErrTrig`

Example See example for `AP.S1DSP.Bittest.ChAErrRdg`.

AP.S1DSP.Bittest.ChAErrTrig

Method

Syntax **`AP.S1DSP.Bittest.ChAErrTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.Bittest.ChAErrRdg` command. The reading in progress is aborted.

See Also `AP.S1DSP.Bittest.ChAErrRdg`, `AP.S1DSP.Bittest.ChAErrReady`

Example See example for `AP.S1DSP.Bittest.ChAErrRdg`.

AP.S1DSP.Bittest.ChAOutput

Property

Syntax **`AP.S1DSP.Bittest.ChAOutput`**

Data Type Boolean

`True`	ON.
`False`	OFF.

Description This command sets Digital Data Analyzer Generator Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.Bittest.ChBOutput`

Example Output See example for `AP.S1DSP.Bittest.Ampl`.

AP.S1DSP.Bittest.ChBDataRdg

Property

Syntax **AP.S1DSP.Bittest.ChBDataRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: dec.

Description This command returns a unsettled reading for the Digital Data Analyzer channel B Data meter and zeros the ready count.

See Also AP.S1DSP.Bittest.ChBDataReady, AP.S1DSP.Bittest.ChBDataTrig

Example

```
Sub Main
   AP.File.OpenTest “BITTEST1.AT1"  ‘Open test
   AP.S1DSP.Bittest.Output = True
   AP.S1DSP.Bittest.Ampl(“FFS”) = .995
   AP.S1DSP.Bittest.ChBOutput = True
   AP.S1DSP.Bittest.ChBDataTrig  ‘Trigger a new reading
   Do
      Ready = AP.S1DSP.Bittest.ChBDataReady
   Loop Until Ready > 0          ‘Wait for new reading
   Reading1 = AP.S1DSP.Bittest.ChBDataRdg(“dec”) ‘Get _
      new reading
   NewLine$ = Chr(13)
   a$= “CH 2 Data  ”+Left(Str$(Reading1),6)+"dec"
   AP.Prompt.Text = a$ + NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.Bittest.ChBDataReady

Property

Syntax **AP.S1DSP.Bittest.ChBDataReady**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Data Analyzer channel B Data 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 `AP.S1DSP.Bittest.ChBDataRdg` or `AP.S1DSP.Bittest.ChBDataTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.Bittest.ChBDataRdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.Bittest.ChBDataRdg`, `AP.S1DSP.Bittest.ChBDataTrig`

Example See example for `AP.S1DSP.Bittest.ChBDataRdg`.

AP.S1DSP.Bittest.ChBDataTrig

Method

Syntax **`AP.S1DSP.Bittest.ChBDataTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.Bittest.ChBDataRdg` comand. The reading in progress is aborted.

See Also `AP.S1DSP.Bittest.ChBDataRdg`, `AP.S1DSP.Bittest.ChBDataReady`

Example See example for `AP.S1DSP.Bittest.ChBDataRdg`.

AP.S1DSP.Bittest.ChBErrRdg

Property

Syntax **`AP.S1DSP.Bittest.ChBErrRdg(`***`unit$`***`)`**

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: dec.

Description This command returns a unsettled reading for the Digital Data Analyzer channel B Errors meter and zeros the ready count.

See Also AP.S1DSP.Bittest.ChBErrReady, AP.S1DSP.Bittest.ChBErrTrig

Example

```
Sub Main
   AP.File.OpenTest “BITTEST1.AT1"     ‘Open test
   AP.S1DSP.Bittest.Output = True
   AP.S1DSP.Bittest.Ampl(“FFS”) = .995
   AP.S1DSP.Bittest.ChAOutput = True
   AP.S1DSP.Bittest.ChBOutput = True
   AP.S1DSP.Bittest.DisplayError = 1   ‘Set Error _
      Display Maximum
   AP.S1DSP.Bittest.RdgRate = 0 ‘Set Reading Rate Auto
   AP.S1DSP.Bittest.ChBErrTrig  ‘Trigger a new reading
   Do
      Ready = AP.S1DSP.Bittest.ChBErrReady
   Loop Until Ready > 0       ‘Wait for new reading
   Reading1 = AP.S1DSP.Bittest.ChBErrRdg(“dec”) ‘Get _
      new reading
   NewLine$ = Chr(13)
   a$= “Ch 2 Errors ”+Left(Str$(Reading1),6)+"dec"
   AP.Prompt.Text = a$ + NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.Bittest.ChBErrReady

Property

Syntax **AP.S1DSP.Bittest.ChBErrReady**

Data Type Integer

0	Reading not ready.

>0	Reading ready.

Description This command returns the Digital Data Analyzer channel B Errors 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 `AP.S1DSP.Bittest.ChBErrRdg` or `AP.S1DSP.Bittest.ChBErrTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.Bittest.ChBErrRdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.Bittest.ChBErrRdg`, `AP.S1DSP.Bittest.ChBErrTrig`

Example See example for `AP.S1DSP.Bittest.ChBErrRdg`.

AP.S1DSP.Bittest.ChBErrTrig

Method

Syntax **`AP.S1DSP.Bittest.ChBErrTTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.Bittest.ChBErrRdg` command. The reading in progress is aborted.

See Also `AP.S1DSP.Bittest.ChBErrTRdg`, `AP.S1DSP.Bittest.ChBErrTReady`

Example See example for `AP.S1DSP.Bittest.ChBErrRdg`.

AP.S1DSP.Bittest.ChBOutput

Property

Syntax **`AP.S1DSP.Bittest.ChBOutput`**

Data Type Boolean

True	ON.
False	OFF.

Description This command sets the Digital Data Analyzer Generator Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.Bittest.ChAOutput`

Example See example for `AP.S1DSP.Bittest.ChBDataRdg`.

AP.S1DSP.Bittest.ConstantValue

Property

Syntax **`AP.S1DSP.Bittest.ConstantValue(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: dec, hex

Description This command sets the Digital Data Analyzer Constant Waveform Value.

The Constant mode generates a continuous stream of data samples at the same fixed value. This is the digital equivalent of a DC voltage source. The value may be entered in decimal or hexadecimal notation. The `AP.S1DSP.Bittest.Freq` and `AP.S1DSP.Bittest.Ampl` commands have no effect with Constant waveform.

The Value entered in Constant mode is with respect to the word length entered in the Resolution field `AP.S1Dio.Resolution` of the Digital I/O panel. Entering a Value of 1 with Resolution of 16 bits, for example, produces a constant stream of samples with the binary value 0000 00 00 0000 0001. (Actually, a 24-bit word is generated with the remaining eight least bits set to zero, so the actual binary output word would be 0000 0000 000 0 0001 00 00 0000. Only the 16 most significant bits (MSBs) would normally be connected to a 16-bit device under test.) If the Resolution field is changed to 24 with the Value

remaining at decimal 1, the binary output word would now be 0000 000 0 0000 00 00 0000 0001.

See Also AP.S1DSP.Bittest.Wfm

Example

```
Sub Main
   AP.File.OpenTest "BITTEST1.AT1"  'Open test
   AP.S1DSP.BitTest.Wfm = 0         'Set Waveform to _
      Constant
   AP.S1DSP.BitTest.DisplayError = 0 'Set Error _
      Display to Normal
   AP.S1DSP.BitTest.RdgRate = 0  'Set Reading Rate to _
      Auto
   AP.S1DSP.BitTest.WfmDisplay = 0  'Set Wave Display _
      to Interpolate
   AP.S1DSP.BitTest.DataInvalid = True
   AP.S1DSP.BitTest.FreezeOnError = False
   AP.S1DSP.BitTest.ConstantValue("hex") = &H5555
   Wait 1
   Reading1 = AP.S1DSP.BitTest.ChADataRdg("dec")
   Reading2 = AP.S1DSP.BitTest.ChBDataRdg("dec")
   NewLine$ = Chr(13)
   a$= "Ch 1 Data "+Left(Str$(Reading1),6)+"dec"
   b$= "Ch 2 Data "+Left(Str$(Reading2),6)+"dec"
   AP.Prompt.Text = a$ + NewLine$ + b$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
   Reading1 = AP.S1DSP.BitTest.ChAErrRdg("dec")
   Reading2 = AP.S1DSP.BitTest.ChBErrRdg("dec")
   NewLine$ = Chr(13)
   a$= "Ch 1 Errors "+Left(Str$(Reading1),6)+"dec"
   b$= "Ch 2 Errors "+Left(Str$(Reading2),6)+"dec"
   AP.Prompt.Text = a$ + NewLine$ + b$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.Bittest.DataInvalid

Property

Syntax `AP.S1DSP.Bittest.DataInvalid`

Data Type Boolean

True	Set.
False	Cleared.

Description This command sets or clears the validity bit.

The command returns a TRUE if the validity bit is Set and FALSE if the validity bit is cleared.

The validity (V) bit on the AES/EBU and SPDIF/EIAJ interfaces may be set or cleared as desired to test the behavior of equipment in response to this bit. When Send Data Invalid Bit checkbox is checked (Set), the V bit is asserted (Invalid). When the box is not checked (Cleared), the V bit is not asserted (Valid). Some equipment is designed to mute when the Invalid condition appears at its digital input. Digital recorders commonly set their output bit to Invalid when the tape is not playing and to Valid when the tape is in motion. The selected validity bit will be generated in every sample until changed on the Digital Data Analyzer panel.

Example See example for `AP.S1DSP.Bittest.ConstantValue`.

AP.S1DSP.Bittest.DisplayError

Property

Syntax `AP.S1DSP.Bittest.DisplayError`

Data Type Integer

0	Normal
1	Maximum.
2	Totalize.

Description This command sets the mode for the Digital Data Analyzer channel A and B Error displays.

Received data is also measured to determine if it matches the data transmitted. Only the number of bits selected in the Resolution field

AP.S1Dio.Resolution of the Digital I/O panel will be analyzed. This comparison is done with algorithms which are insensitive to delay between the send and receive sections. The number of errors in the received data per measurement interval are counted for each channel. The AP.S1DSP.Bittest.DisplayError command selects the type of analysis to be performed. In the Normal mode, the number of errors detected during the last measurement interval (1/4 second in Slow or 1/32 second in Fast Reading Rate) are displayed directly in the Ch 1 and Ch 2 Errors fields of the panel. If Error Display is selected as Maximum, the maximum error count during any measurement interval will be held in the display. A running total of all errors may be accumulated by using the Totalize mode of the Error Display field.

See Also AP.S1DSP.Bittest.RdgRate

Example See example for AP.S1DSP.Bittest.ConstantValue.

AP.S1DSP.Bittest.FreezeOnError

Property

Syntax **AP.S1DSP.Bittest.FreezeOnError**

Data Type Boolean

True	Hold first error reading..
False	Continue updating data readings.

Description This command sets or clears the Freeze Data on Error field on the Digital Data Analyzer.

If the AP.S1DSP.Bittest.FreezeOnError command is set to (True), the Data fields will continue to display the value which was received when the first error occurred. If AP.S1DSP.Bittest.FreezeOnError is set to (False), the Data fields will continue updating, independent of any errors detected.

See Also AP.S1DSP.Bittest.RdgRate

Example See example for AP.S1DSP.Bittest.ConstantValue.

AP.S1DSP.Bittest.Freq

Property

Syntax `AP.S1DSP.Bittest.Freq(unit$)`

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following unit is valid for this command: Hz

Description This command sets the Digital Data Analyzer Digital Generator frequency.

Example Output See example for `AP.S1DSP.Bittest.Ampl`.

AP.S1DSP.Bittest.Output

Property

Syntax `AP.S1DSP.Bittest.Output`

Data Type Boolean

True	On
False	Off

Description This command sets the Digital Data Analyzer channel A and B outputs to ON or OFF if they have been individually enabled by the `AP.S1DSP.Bittest.ChAOutput` and `AP.S1DSP.Bittest.ChBOutput` commands.

See Also `AP.S1DSP.Bittest.ChAOutput`, `AP.S1DSP.Bittest.ChBOutput`

Example Output See example for `AP.S1DSP.Bittest.Ampl`.

AP.S1DSP.Bittest.RdgRate

Property

Syntax `AP.S1DSP.Bittest.RdgRate`

Data Type Integer

0	Auto reading rate. The reading rate is automatically selected based on the measured frequency.
1	Slow 1/4 second update rate.
2	Fast 1/32 second update rate.

Description This command sets the rate a which the Data (and Errors) readings are updated.

Example See example for `AP.S1DSP.Bittest.ConstantValue`.

AP.S1DSP.Bittest.Wfm

Property

Syntax `AP.S1DSP.Bittest.Wfm`

Data Type Integer

0	Constant
1	Random.
2	Walking-1.
3	Walking-0.
4	Sine.

Description This command sets the Digital Data Analyzer Generator Waveform.

Example See example for `AP.S1DSP.Bittest.ConstantValue`.

AP.S1DSP.Bittest.WfmDisplay

Property

Syntax `AP.S1DSP.Bittest.WfmDisplay`

Data Type Integer

0	Interpolate
1	Display Samples.
2	Peak Values.
3	Absolute Values.

Description

This command sets the Digital Data 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 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 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 thus always sends a positive number to the computer. The advantage of Absolute Values mode is that logarithms may be computed when all numbers involved are positive, so a dB units may be used on the Y axis to display the waveform. Waveform display with Absolute Values mode thus can create a wide dynamic range oscilloscope which displays the envelope of an audio signal, calibrated in familiar dB units such as dBV, dBm, 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 `AP.S1DSP.Bittest.ConstantValue`.

User Notes

User Notes

User Notes

Codec Tester

AP.S1DSP.Codec.Ampl

Property

Syntax **AP.S1DSP.Codec.Ampl(*unit$*)**

Data Type Double — Valid amplitude settings are from 0 to 100 %FS.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits

Description This command sets the Codec Tester Generator Amplitude.

Example

```
Sub Main
   AP.File.OpenTest "CODECA.AT1" 'Open test
   AP.S1DSP.Codec.Ampl("FFS") = .95
   AP.S1DSP.Codec.Ch1Source = 4
   AP.S1DSP.Codec.Ch2Source = 4
   Wait 1
   AP.S1DSP.Codec.Ch1Trig
   Do
      Ready1 = AP.S1DSP.Codec.Ch1Ready
   Loop Until Ready1 > 0
   Reading1 = AP.S1DSP.Codec.Ch1Rdg("FFS")
   AP.S1DSP.Codec.Ch2Trig
   Do
      Ready2 = AP.S1DSP.Codec.Ch2Ready
   Loop Until Ready2 > 0
   Reading2 = AP.S1DSP.Codec.Ch2Rdg("FFS")
   NewLine$ = Chr(13)
   a$= "Ch 1 Source "+Left(Str$(Reading1),6)+"FFS"
   b$= "Ch 2 Source "+Left(Str$(Reading2),6)+"FFS"
   AP.Prompt.Text = a$ + NewLine$ + b$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.Codec.Ch1Rdg

Property

Syntax `AP.S1DSP.Codec.Ch1Rdg(unit$)`

Data Type Variant

Parameters

Part	Description
unit$	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 Codec Tester channel 1 Peak Monitor meter and zeros the ready count.

See Also `AP.S1DSP.Codec.Ch1Ready, AP.S1DSP.Codec.Ch1Trig`

Example See example for `AP.S1DSP.Codec.Ampl`.

AP.S1DSP.Codec.Ch1Ready

Property

Syntax `AP.S1DSP.Codec.Ch1Ready`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Codec Tester channel 1 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 `AP.S1DSP.Codec.Ch1Rdg` or `AP.S1DSP.Codec.Ch1Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.Codec.Ch1Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.Codec.Ch1Rdg, AP.S1DSP.Codec.Ch1Trig`

Example See example for `AP.S1DSP.Codec.Ampl`.

AP.S1DSP.Codec.Ch1Source

Property

Syntax `AP.S1DSP.Codec.Ch1Source`

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.Codec.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Gen-Mon
5	DSP A
6	DSP B
7	None

The following list contains the selections relevant to the `AP.S1DSP.Codec.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Codec Tester Channnel 1 Input Source.

Example

```
Sub Main
   If AP.Application.SysType = “2" Then
      AP.Prompt.Text = ”System One procedure only ..."
      AP.Prompt.ShowWithContinue
      End
   End If
   AP.Application.NewTest      ‘Initialize APWIN
   AP.Log.Enable = 0           ‘Disable log file

   ‘Set up Analog Generator
   AP.Gen.Wfm 17               ‘DGEN
   AP.Gen.Output = True        ‘Output ON
   ‘Set up Analog Analyzer
```

```
With AP.Anlr
   .ChARangeAuto = False   'Turn off autorange
   .ChBRangeAuto = False   'Turn off autorange
   .ChARange("V") = 1.2    'Set fixed range
   .ChBRange("V") = 1.2    'Set fixed range
   .FuncMode = 9           '2-Channel Amplitude
End With

'Set up DSP
AP.S1Dsp.Program = 9       'CODEC
With AP.S1Dsp.Codec
   .InputFormat = 0        'A/D
   .Mode = 4               'Masking
   .Ch1Source = 0          'Anlr-A
   .Ch2Source = 1          'Anlr-B
   .FreqRes("%") = 0.0
   .Window = 0             'None
   .FreqCorrection = False 'Off
   .Warnings = False       'Off
   .TrigCriteria = 0       'Normal
   .TrigSource = 8         'Free Run
   .WfmName = "DISTOR48.AGM"
   .ChAOutput = True       'Turn Ch A on
   .ChBOutput = True       'Turn Ch B on
   .Output = True 'Turn Both A & B on (muting off)
End With

'Set up Sweep
With AP.Sweep
   .Data1.Id = 6333        'Codec.Ch.1 Ampl
   .Data1.Top("dBV") = 0.0
   .Data1.Bottom("dBV") = -150.0
   .Source1.Id = 5630      'Codec.FFT Freq
   .Source1.Start("Hz") = 17.5
   .Source1.Stop("Hz") = 20e3
   .Source1.Steps = 500
   .Append = True          'Append all waveforms
   .Stereo = True 'Ch 1 and Ch 2
   .CreateGraph = True     'Graph all waveforms
   .Start                  'Start sweep.
   AP.File.SaveDataAs "MASK.ADL"    'Save masking _
```

```
        curve as MASK.ADL limit file.
    AP.Application.NewData  'Remove Masking curve _
        data from memory.
    AP.S1Dsp.Codec.Mode = 0 'spectrum mode.
    .Reprocess  'Reprocess the acquired waveform _
        and display spectrum results.
    .Source1.Table ("CODEC.ADS",0)   'Attach Sweep _
        table.
    AP.S1Dsp.Codec.Mode = 1 'Select response mode.
    .Reprocess  'Reprocess the acquired waveform and _
        display response results.
    .Data1.Limits ("MASK.ADL", 1, True) 'Attach upper _
        limits to Data1 & Data3.
    .Data3.Limits ("MASK.ADL", 1, True)
    AP.S1Dsp.Codec.Mode = 2 'Select distortion mode.
    .Reprocess  'Reprocess the acquired waveform _
        and display distortion results.
    AP.S1Dsp.Codec.Mode = 3 'Select noise mode.
    .Reprocess  'Reprocess the acquired waveform _
        and display noise results.
End WithEnd Sub
```


AP.S1DSP.Codec.Ch1Trig

Method

Syntax `AP.S1DSP.Codec.Ch1Trig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.Codec.Ch1Rdg` command. The reading in progress is aborted.

See Also `AP.S1DSP.Codec.Ch1Rdg, AP.S1DSP.Codec.Ch1Ready`

Example See example for `AP.S1DSP.Codec.Ampl`.

AP.S1DSP.Codec.Ch2Rdg

Property

Syntax `AP.S1DSP.Codec.Ch2Rdg(`*unit$*`)`

Data Type Variant

Parameters

Part	Description
unit$	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 Codec Tester channel 2 Peak Monitor meter and zeros the ready count.

See Also `AP.S1DSP.Codec.Ch2Ready, AP.S1DSP.Codec.Ch2Trig`

Example See example for `AP.S1DSP.Codec.Ampl`.

AP.S1DSP.Codec.Ch2Ready

Property

Syntax **`AP.S1DSP.Codec.Ch2Ready`**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Codec Tester channel 2 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 `AP.S1DSP.Codec.Ch2Rdg` or `AP.S1DSP.Codec.Ch2Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.Codec.Ch2Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.Codec.Ch2Rdg, AP.S1DSP.Codec.Ch2Trig`

Example See example for `AP.S1DSP.Codec.Ampl`.

AP.S1DSP.Codec.Ch2Source

Property

Syntax `AP.S1DSP.Codec.Ch1Source`

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.Codec.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Gen-Mon
5	DSP A
6	DSP B
7	None

The following list contains the selections relevant to the `AP.S1DSP.Codec.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Codec Tester Channnel 2 Input Source.

Example See example for `AP.S1DSP.Codec.Ch1Source`.

AP.S1DSP.Codec.Ch2Trig

Method

Syntax `AP.S1DSP.Codec.Ch2Trig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.Codec.Ch2Rdg` comand. The reading in progress is aborted.

See Also `AP.S1DSP.Codec.Ch2Rdg, AP.S1DSP.Codec.Ch2Ready`

Example See example for `AP.S1DSP.Codec.Ampl`.

AP.S1DSP.Codec.ChAOutput

Property

Syntax **`AP.S1DSP.Codec.ChAOutput`**

Data Type Boolean

`True`	ON.
`False`	OFF.

Description This command sets Codec Tester Generator channel A Output to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.Codec.ChBOutput`

Example See example for `AP.S1DSP.Codec.Ch1Source`.

AP.S1DSP.Codec.ChBOutput

Property

Syntax **`AP.S1DSP.Codec.ChBOutput`**

Data Type Boolean

`True`	ON.
`False`	OFF.

Description This command sets the Digital Data Analyzer Generator channel B Output to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.Codec.ChAOutput`

Example See example for `AP.S1DSP.Codec.Ch1Source`.

AP.S1DSP.Codec.FreqCorrection

Property

Syntax `AP.S1DSP.Codec.FreqCorrection`

Data Type Boolean

True	Frequency correction ON.
False	Frequency correction OFF.

Description This command enables or disables frequency error correction on acquired waveform data.

A key feature of CODEC 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 CODEC 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 than the analyzer or due to analog tape player speed errors), CODEC 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%.

See Also `AP.S1DSP.Codec.FreqRes`

Example See example for `AP.S1DSP.Codec.Ch1Source`.

AP.S1DSP.Codec.FreqRes

Property

Syntax `AP.S1DSP.Codec.FreqRes(unit$)`

Data Type Double — Valid amplitude settings are from +/- 0.0 to 13.0 %.

Parameters

Name	Description
unit$	String that designates the desired unit. The following unit is valid for this command: %

Description This command sets the Codec Tester Frequency Resolution.

The Frequency Resolution field is a numeric entry field with % units. The user may enter a value up to 13%, which is used in the Response and Distortion Measurement modes.

In the Response mode, 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 the Distortion mode, 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 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 `AP.S1DSP.Codec.Ch1Source`.

AP.S1DSP.Codec.InputFormat

Property

Syntax `AP.S1DSP.Codec.InputFormat`

Data Type Integer

0	A/D
1	Digital.

Description This command sets the Codec Tester Input Format.

Example See example for `AP.S1DSP.Codec.Ch1Source`.

AP.S1DSP.Codec.Mode

Property

Syntax `AP.S1DSP.Codec.Mode`

Data Type Integer

0 Spectrum: this selection 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 in use results in more FFT bins in the Start-Stop frequency span being plotted than the number of Steps, 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.

1 Response: this selection is always used with a sweep table (.ADS file) which lists the exact frequencies of the sinewaves in the multitone signal which are to be used for frequency response measurements. The DSP sends to the computer only the amplitudes of the FFT bins containing those exact frequencies, resulting in a frequency response graph. There are typically from 3 to 30 sinewaves in most multitone signals.

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 tones 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 of CODEC combines this spread energy back into a single value, much as the human hearing system responds to signals with small amounts of FM.

2 Distortion: this selection 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 thus 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 (see below).

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 tones 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. This is similar to how human hearing system masks low amplitude signals near to a stronger signal.

3 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 CODEC Transform length being set to a value that is 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.

4 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 CODEC 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.

Description This command sets the Codec Tester measurement mode. The `AP.S1DSP.Codec.Mode` command controls the type of post-processing done to FFT results.

Example See example for `AP.S1DSP.Codec.Ch1Source`.

AP.S1DSP.Codec.Output

Property

Syntax `AP.S1DSP.Codec.Output`

Data Type Boolean

True	On
False	Off

Description This command sets the Codec Tester channel A and B outputs to ON or OFF if they have been individually enabled by the `AP.S1DSP.Codec.ChAOutput` and `AP.S1DSP.Codec.ChBOutput` commands.

See Also `AP.S1DSP.Codec.ChAOutput, AP.S1DSP.Codec.ChBOutput`

Example See example for `AP.S1DSP.Codec.Ch1Source`.

AP.S1DSP.Codec.TrigCriteria

Property

Syntax `AP.S1DSP.Codec.TrigCriteria`

Data Type Integer

0	Normal
1	Tight
2	Loose

Description This command sets the Codec Tester Trigger Criteria.

To permit user control of the triggering criteria, the allowable deviation from reference signal amplitude at generator tone frequencies and the amount that energy at all other frequencies must be attenuated are settable at three values. The Triggering Criteria field allows the user to select Tight, Normal, and Loose conditions. Each selection 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 CODEC 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 `AP.S1DSP.Codec.Ch1Source`.

AP.S1DSP.Codec.TrigSource

Property

Syntax `AP.S1DSP.Codec.TrigSource`

Data Type Integer

0	Auto + 0 ms
1	Auto +100 ms
2	Auto + 200 ms
3	Auto + 400 ms
4	Auto + 1 sec
5	Auto + 2 sec

All Auto selections cause triggering (after the stated time delay) if either channel 1 or channel 2 has a signal amplitude greater than digital zero. These are the normal operating modes of CODEC, with the zero delay selection requiring the shortest signal duration.

6	Dgen: this selection functions only on Dual Domain units (SYS-300 series). If CODEC is generating a signal from a waveform file, a Digital Generator trigger is issued each time the first sample from the file is generated.
7	External: this source is operational only with SYS-322 (Dual Domain) units. It 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 acquisiton being triggered on the next sample after pin 3 is pulled high.

8	Free Run: when selected, signal acquisition begins immediately after F9, `AP.Sweep.Start,` or Go is initiated, regardless of signal amplitude.

Description This command sets the Codec Tester Trigger Source & Delay.

In-service multitone testing of broadcast transmission circuits and other similar applications involves inserting a brief burst of multitone signal into a short pause in program material. Broadcast transmission circuits typically include modulation processors and other forms of compressors and limiters. It may be desirable to allow a certain stabilization time following the start of the multitone signal before the measurement is made, in order that the measured conditions represent something approaching steady-state rather than initial transient conditions. The `AP.S1DSP.Codec.TrigSource` command permits control of this stabilization time. The duration of multitone signal transmitted must also be lengthened when non-zero values of delay are used.

Example See example for `AP.S1DSP.Codec.Ch1Source`.

AP.S1DSP.Codec.Warnings

Property

Syntax **`AP.S1DSP.Codec.Warnings`**

Data Type Boolean

True	Warnings ON.
False	Warnings OFF.

Description This command enables or disables frequency error correction warnings on Digital Generator waveform data.

When a Digital Generator waveform file is first loaded, CODEC analyzes the multitone signal described in the waveform file and determines whether it contains enough sinewaves across the spectrum above 200 Hz, at sufficient frequency spacing, to provide reliable frequency error correction on an acquired signal. If the `AP.S1DSP.Codec.Warnings` command is set to True (ON), a warning message will be displayed whenever a waveform file is loaded

which does not meet the built-in criteria. In many cases, the built-in criteria are somewhat conservative and signals with fewer tones or closer spacing may still work properly. Since display of a warning message will interrupt the flow of the program. Setting the `AP.S1DSP.Codec.Warnings` command to False (OFF) provides a means of ignoring the message and thus avoiding hang-up of the program when it has been experimentally determined that frequency error correction works properly with a particular signal.

See Also `AP.S1DSP.Codec.FreqCorrection`

Example See example for `AP.S1DSP.Codec.Ch1Source`.

AP.S1DSP.Codec.WfmName

Method

Syntax **`AP.S1DSP.Codec.WfmName =`** *`filename$`*

Data Type Boolean

Parameters

Name	Description
`filename$`	Any valid DOS path and file name. The file must be an APWIN waveform file (.agm or .ags).

Description This command loads the designated waveform file (.AGM or .AGS) into the Multitone Generator/Analyzer Digital Generator.

AP.S1DSP.Codec.Window

Property

Syntax **`AP.S1DSP.Codec.Window`**

Data Type Integer

0	None
1	Hann

Description This command sets the Multitone Generator/Analyzer Window selection. See Appendix E for FFT Window Discriptions.

Example See example for `AP.S1DSP.Codec.Ch1Source`.

User Notes

User Notes

User Notes

User Notes

Multitone Generator/Analyzer

AP.S1DSP.FastTest.Ampl

Property

Syntax **AP.S1DSP.FastTest.Ampl(*unit$*)**

Data Type Double — Valid amplitude settings are from 0 to 100 %FS.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits

Description This command sets the Multitone Generator/Analyzer Digital Generator Amplitude.

Example

```
Sub Main
   AP.File.OpenTest "Fasttst1.at1"  'Opens test
   With AP.S1Dsp.FastTest
      .WfmName = "Iso31.agm"  'Load Waveform file
      .Ampl("%FS") = -.05     'Set DGEN amplitude
      .ChAOutput = True    'Set Ch A DGEN on
      .ChBOutput = False   'Set Ch B DGEN off
      .Output = True       'Set DGEN output on
      .InputFormat = 0     'Set Input to A/D
      .Ch1Source = 0       'Set Source Anlr A
      .FFTLength = 0       'Set FFT Length to Max
      .Window = 0          'Set FFT Window to None
      .Mode = 0            'Set Measurement Mode to _
         Spectrum
      .FreqRes("%") = .1   'Set Freq Res to .1%
      .PhaseDisplay = 0    'Set Phase Display to _
         Independent
      .TrigSource = 4      'Set Triggering to DGEN
      AP.Sweep.Start
      AP.Sweep.Source1.Table("Fasttest.ads", 0)
      .Mode = 1            'Set Measurement Mode to _
         Response
      AP.Sweep.Reprocess
      .Mode = 2            'Set Measurement Mode to _
         Distortion
```

```
      AP.Sweep.Reprocess
      .Mode = 3            'Set Measurement Mode to _
         Spectrum
      AP.Sweep.Reprocess
   End With
End Sub
```

Comment This macro uses a multitone signal To display the waveform spectrum, requency, distortion, and noise responses.

AP.S1DSP.FastTest.Ch1Rdg

Property

Syntax **AP.S1DSP.FastTest.Ch1Rdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	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 Generator/Analyzer channel 1 Peak Monitor meter and zeros the ready count.

See Also AP.S1DSP.FastTest.Ch1Ready, AP.S1DSP.FastTest.Ch1Trig

Example

```
Sub Main
   AP.File.OpenTest "FASTTST2.AT1"     'Opens test
   AP.S1DSP.FastTest.ChAOutput = True
   AP.S1DSP.FastTest.Output = True
   AP.S1DSP.FastTest.InputFormat = 0   'Set Input to A/D
   AP.S1DSP.FastTest.Ch1Source = 0
      Wait 1
   AP.S1DSP.FastTest.Ch1Trig     'Trigger a new reading
   Do
      Ready = AP.S1DSP.FastTest.Ch1Ready
   Loop Until Ready > 0       'Wait for new reading
   Reading1 = AP.S1DSP.FastTest.Ch1Rdg("FFS") 'Get new _
      reading
   NewLine$ = Chr(13)
```

```
   a$= "Ch 1 Peak Mon "+Left(Str$(Reading1),6)+"FFS"
   AP.Prompt.Text = a$ + NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.FastTest.Ch1Ready

Property

Syntax `AP.S1DSP.FastTest.Ch1Ready`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Multitone Generator/Analyzer channel 1 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 `AP.S1DSP.FastTest.Ch1Rdg` or `AP.S1DSP.FastTest.Ch1Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.FastTest.Ch1Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.FastTest.Ch1Rdg`, `AP.S1DSP.FastTest.Ch1Trig`

Example See example for `AP.S1DSP.FastTest.Ch1Rdg`.

AP.S1DSP.FastTest.Ch1Source

Property

Syntax `AP.S1DSP.FastTest.Ch1Source`

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.FastTest.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Gen-Mon
5	DSP A
6	DSP B
7	None

The following list contains the selections relevant to the `AP.S1DSP.FastTest.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Multitone Generator/Analyzer Channnel 1 Input.

Example See example for `AP.S1DSP.FastTest.Ch1Rdg`.

AP.S1DSP.FastTest.Ch1Trig

Method

Syntax `AP.S1DSP.FastTest.Ch1Trig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.FastTest.Ch1Rdg` command. The reading in progress is aborted.

See Also `AP.S1DSP.FastTest.Ch1Rdg`, `AP.S1DSP.FastTest.Ch1Ready`

Example See example for `AP.S1DSP.FastTest.Ch1Rdg`.

AP.S1DSP.FastTest.Ch2Rdg

Property

Syntax **`AP.S1DSP.FastTest.Ch2Rdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description
`unit$`	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 Generator/Analyzer channel 2 Peak Monitor meter and zeros the ready count.

See Also `AP.S1DSP.FastTest.Ch2Ready`, `AP.S1DSP.FastTest.Ch2Trig`

Example

```
Sub Main
   AP.File.OpenTest “FASTTST2.AT1"     ‘Open test
   AP.S1DSP.FastTest.ChBOutput = True
   AP.S1DSP.FastTest.Output = True
   AP.S1DSP.FastTest.InputFormat = 0 ‘Set Input to A/D
   AP.S1DSP.FastTest.Ch2Source = 0
   Wait 1
   AP.S1DSP.FastTest.Ch2Trig     ‘Trigger a new reading
   Do
      Ready = AP.S1DSP.FastTest.Ch2Ready
   Loop Until Ready > 0          ‘Wait for new reading
   Reading1 = AP.S1DSP.FastTest.Ch2Rdg(“FFS”)‘Get new _
      reading
   NewLine$ = Chr(13)
   a$= “Ch 2 Peak Mon ”+Left(Str$(Reading1),6)+"FFS"
```

```
    AP.Prompt.Text = a$ + NewLine$
    AP.Prompt.ShowWithContinue
    Beep
    Stop
End Sub
```

AP.S1DSP.FastTest.Ch2Ready

Property

Syntax `AP.S1DSP.FastTest.Ch2Ready`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Multitone Generator/Analyzer channel 2 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 `AP.S1DSP.FastTest.Ch2Rdg` or `AP.S1DSP.FastTest.Ch2Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.FastTest.Ch2Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.FastTest.Ch2Rdg`, `AP.S1DSP.FastTest.Ch2Trig`

Example See example for `AP.S1DSP.FastTest.Ch2Rdg`.

AP.S1DSP.FastTest.Ch2Source

Property

Syntax `AP.S1DSP.FastTest.Ch2Source`

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.FastTest.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Gen-Mon
5	DSP A
6	DSP B
7	None

The following list contains the selections relevant to the `AP.S1DSP.FastTest.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Multitone Generator/Analyzer Channnel 2 Input.

Example See example for `AP.S1DSP.FastTest.Ch2Rdg`.

AP.S1DSP.FastTest.Ch2Trig

Method

Syntax **`AP.S1DSP.FastTest.Ch2Trig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.FastTest.Ch2Rdg` comand. The reading in progress is aborted.

See Also `AP.S1DSP.FastTest.Ch2Rdg`, `AP.S1DSP.FastTest.Ch2Ready`

Example See example for `AP.S1DSP.FastTest.Ch2Rdg`.

AP.S1DSP.FastTest.ChAOutput

Property

Syntax **`AP.S1DSP.FastTest.ChAOutput`**

Data Type Boolean

True	ON.
False	OFF.

Description This command sets Multitone Generator/Analyzer Generator Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.FastTest.ChBOutput`

Example See example for `AP.S1DSP.FastTest.Ampl`.

AP.S1DSP.FastTest.ChBOutput

Property

Syntax **`AP.S1DSP.FastTest.ChBOutput`**

Data Type Boolean

True	ON.
False	OFF.

Description This command sets the Multitone Generator/Analyzer Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.FastTest.ChAOutput`

Example See example for `AP.S1DSP.FastTest.Ampl`.

AP.S1DSP.FastTest.FFTLength

Property

Syntax `AP.S1DSP.FastTest.FFTLength`

Data Type Integer

0	Maximum
1	4096
2	2048
3	1024
4	512
5	256

Description

This command sets the Multitone Generator/Analyzer FFT Length.

Maximum means 16384 on all System One Dual Domain units and on System One Plus DSP units with the MEM (memory) option, but is the same as the 4096 selection on System One Plus DSP units without the MEM option.

The `AP.S1DSP.FastTest.FFTLength` command controls the record length (number of words of memory, in samples) which will be filled when the `AP.Sweep.Start` (F9/Go) command is executed. The FFT Length field value also controls the record length used as input to the FFT process when either `AP.Sweep.Start` (F9/Go) is initiated to acquire and transform, or the `AP.Sweep.Reprocess` (F6 function key) is used to re-transform a record previously acquired. Longer transform lengths produce greater frequency resolution in the resulting FFT, but require longer times to acquire and to transform the signal.

AP.S1DSP.FastTest.FreqRes

Property

Syntax `AP.S1DSP.FastTest.FreqRes(`*`unit$`*`)`

Data Type Double — Valid amplitude settings are from +/- 0.0 to 13.0 %.

Parameters

Name	Description

unit$	String that designates the desired unit. The following unit is valid for this command: %

Description This command sets the Multitone Generator/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 modes.

In Response mode, 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 mode, 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 `AP.S1DSP.FastTest.Ampl`.

AP.S1DSP.FastTest.InputFormat

Property

Syntax **`AP.S1DSP.FastTest.InputFormat`**

Data Type Integer

0	A/D
1	Digital

Description This command sets the Multitone Generator/Analyzer Input Format.

Example See example for `AP.S1DSP.FastTest.Ampl`.

AP.S1DSP.FastTest.Mode

Property

Syntax `AP.S1DSP.FastTest.Mode`

Data Type Integer

0 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.

1 Response: this selection 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 only the amplitudes of the FFT bins containing those exact frequencies, resulting in a frequency response graph. There are typically from 3 to 30 sinewaves in most multitone signals.

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 tones 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

of CODEC combines this spread energy back into a single value, much as the human hearing system responds to signals with small amounts of FM.

2 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 thus 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 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 mode.

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 like 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.

3 Noise: 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 Transform length being set to the value twice the length of the waveform file which generates the multitone signal. The analyzer frequency

resolution is then 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.

Description This command sets the Multitone Generator/Analyzer measurement mode. The `AP.S1DSP.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 `AP.S1DSP.FastTest.Ampl`.

AP.S1DSP.FastTest.Output

Property

Syntax **`AP.S1DSP.FastTest.Output`**

Data Type Boolean

`True`	On
`False`	Off

Description This command sets the Multitone Generator/Analyzer channel A and B outputs to ON or OFF if they have been individually enabled by the `AP.S1DSP.FastTest.ChAOutput` and `AP.S1DSP.FastTest.ChBOutput` commands.

See Also `AP.S1DSP.FastTest.ChAOutput`, `AP.S1DSP.FastTest.ChBOutput`

Example See example for `AP.S1DSP.FastTest.Ampl`.

AP.S1DSP.FastTest.PhaseDisplay

Property

Syntax `AP.S1DSP.FastTest.PhaseDisplay`

Data Type Integer

0	Independent
1	Interchannel

Description This command sets the Multitone Generator/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 thus still plots absolute phase of the channel 1 signal.

Example See example for `AP.S1DSP.FastTest.Ampl`.

AP.S1DSP.FastTest.TrigSource

Property

Syntax `AP.S1DSP.FastTest.TrigSource`

Data Type Integer

0	Free Run
1	Channel #1
2	Channel #2
3	Auto
4	Dgen

Description This command sets the Multitone Generator/Analyzer Trigger Source.

Example See example for `AP.S1DSP.FastTest.Ampl`.

AP.S1DSP.FastTest.WfmName

Method

Syntax **`AP.S1DSP.FastTest.WfmName(`*`filename$`*`)`**

Data Type Boolean

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN waveform file (.agm or .ags).

Description This command loads the designated waveform file (.AGM or .AGS) into the Multitone Generator/Analyzer Digital Generator.

Example See example for `AP.S1DSP.FastTest.Ampl`.

AP.S1DSP.FastTest.Window

Property

Syntax **`AP.S1DSP.FastTest.Window`**

Data Type Integer

0	None
1	Hann
2	Flat-Top
3	Blackman-Harris

Description This command sets the Multitone Generator/Analyzer Window selection. See Appendix E for FFT Window Discriptions.

Example See example for `AP.S1DSP.FastTest.Ampl`.

User Notes

User Notes

User Notes

User Notes

Triggered Multitone Tester

AP.S1DSP.FastTrig.Ampl

Property

Syntax **AP.S1DSP.FastTrig.Ampl(*unit$*)**

Data Type Double — Valid amplitude settings are from 0 to 100 %FS.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits

Description This command sets the Triggered Multitone Tester Digital Generator Amplitude.

Example

```
Sub Main
'This procedure displays readings from Ch1 and Ch2
   AP.File.OpenTest "Fastrg1.at1"   'Open test.
   With AP.S1Dsp.FastTrig
      .Ampl("%FS") = 90 'Set units %FS and level to 90.
      .InputFormat = 0          'Set input to A/D.
      .Ch1Source = 0            'Set Source to Anlr A.
      .Ch2Source = 1            'Set Source to Anlr B.
      .ChAOutput = True         'Set Dgen ChA on.
      .ChBOutput = True         'Set Dgen ChB on.
      .Output = True            'Set Digital Gen on.
         Wait 1.5
      .Ch1Trig                  'Trigger a new reading.
      Do
         Ready1 = .Ch1Ready
         Loop Until Ready1 > 0   'Wait for new reading.
      Reading1 = .Ch1Rdg("FFS")  'Get new reading.
      .Ch2Trig
      Do
         Ready2 = .Ch2Ready
         Loop Until Ready2 > 0
      Reading2 = .Ch2Rdg("FFS")
   End With
   NewLine$ = Chr(13)
   a$= "Ch1 Source "+Left(Str$(Reading1),6)+"FFS"
```

```
    b$= "Ch2 Source "+Left(Str$(Reading2),6)+"FFS"
    AP.Prompt.Text = a$ + NewLine$ + b$ + NewLine
    AP.Prompt.ShowWithContinue
    Beep
    Stop
End Sub
```

Comment `This macro displays readings from Ch1 and Ch2.`

AP.S1DSP.FastTrig.Ch1Rdg

Property

Syntax **`AP.S1DSP.FastTrig.Ch1Rdg(`***`unit$`***`)`**

Data Type Variant

Parameters

Part	Description
unit$	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 Triggered Multitone Tester channel 1 Peak Monitor meter and zeros the ready count.

See Also `AP.S1DSP.FastTrig.Ch1Ready`, `AP.S1DSP.FastTrig.Ch1Trig`

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.Ch1Ready

Property

Syntax **`AP.S1DSP.FastTrig.Ch1Ready`**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Triggered Multitone Tester channel 1 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 `AP.S1DSP.FastTrig.Ch1Rdg` or `AP.S1DSP.FastTrig.Ch1Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.FastTrig.Ch1Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.FastTrig.Ch1Rdg`, `AP.S1DSP.FastTrig.Ch1Trig`

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.Ch1Source

Property

Syntax **`AP.S1DSP.FastTrig.Ch1Source`**

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.FastTrig.InputFormat` command A/D input selection.

0	Anlr-A.
1	Anlr-B.
2	Anlr Reading Amp.
3	Anlr Reading Ratio.
4	Gen-Mon
5	DSP A
6	DSP B
7	None.

The following list contains the selections relevant to the `AP.S1DSP.FastTrig.InputFormat` command Digital input selection.

0	A.
1	B.

2	None.

Description This command sets the Triggered Multitone Tester Channnel 1 Input.

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.Ch1Trig

Method

Syntax **`AP.S1DSP.FastTrig.Ch1Trig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.FastTrig.Ch1Rdg` command. The reading in progress is aborted.

See Also `AP.S1DSP.FastTrig.Ch1Rdg`, `AP.S1DSP.FastTrig.Ch1Ready`

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.Ch2Rdg

Property

Syntax **`AP.S1DSP.FastTrig.Ch2Rdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description
`unit$`	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 Triggered Multitone Tester channel 2 Peak Monitor meter and zeros the ready count.

See Also `AP.S1DSP.FastTrig.Ch2Ready`, `AP.S1DSP.FastTrig.Ch2Trig`

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.Ch2Ready

Property

Syntax `AP.S1DSP.FastTrig.Ch2Ready`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Triggered Multitone Tester channel 2 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 `AP.S1DSP.FastTrig.Ch2Rdg` or `AP.S1DSP.FastTrig.Ch2Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.FastTrig.Ch2Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.FastTrig.Ch2Rdg`, `AP.S1DSP.FastTrig.Ch2Trig`

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.Ch2Source

Property

Syntax `AP.S1DSP.FastTrig.Ch1Source`

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.FastTrig.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl

3	Anlr Reading Ratio
4	Gen-Mon
5	DSP A
6	DSP B
7	None

The following list contains the selections relevant to the `AP.S1DSP.FastTrig.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Triggered Multitone Tester Channnel 2 Input.

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.Ch2Trig

Method

Syntax **`AP.S1DSP.FastTrig.Ch2Trig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.FastTrig.Ch2Rdg` comand. The reading in progress is aborted.

See Also `AP.S1DSP.FastTrig.Ch2Rdg`, `AP.S1DSP.FastTrig.Ch2Ready`

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.ChAOutput

Property

Syntax **`AP.S1DSP.FastTrig.ChAOutput`**

Data Type Boolean

True	ON.
False	OFF.

Description This command sets Triggered Multitone Tester Generator Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.FastTrig.ChBOutput`

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.ChBOutput

Property

Syntax **`AP.S1DSP.FastTrig.ChBOutput`**

Data Type Boolean

True	ON.
False	OFF.

Description This command sets the Triggered Multitone Generator Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.FastTrig.ChAOutput`

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.FreqCorrection

Property

Syntax **`AP.S1DSP.FastTrig.FreqCorrection`**

Data Type Boolean

True	Frequency correction ON.
False	Frequency correction OFF.

Description

This command enables or disables frequency error correction on acquired waveform data.

A key feature of FASTTRIG 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 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, FASTTRIG 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%.

See Also

`AP.S1DSP.FastTrig.FreqRes`

Example

```
Sub Main
   AP.File.OpenTest "Fastrg2.at1"   'Open test
   With AP.S1Dsp.FastTrig
      .Mode = 0   'Set Measurement Mode to Spectrum
      .WfmName = "Iso31.agm"  'Load Waveform Iso31.agm
      .FreqCorrection = True  'Enables Frequency _
         Correction
      .Warnings = True  'Enables Frequency Error _
         Correction Warnings
      .TrigCriteria = 0    'Set Trig Criteria to Normal
      .TrigSource = 6      'Set Trig Source to DGEN
      .FreqRes("%") = 0    'Set FreqRes to 0%
      .Window = 0          'Set FastTrig Window to none
      Wait .5
      AP.Sweep.Start
      'Attach sweep file
      AP.Sweep.Source1.Table("Fasttest.ads", 0)
      .Mode = 1   'Set Measurement Mode to Response
      AP.Sweep.Reprocess
      .Mode = 2   'Set Measurement Mode to Distortion
      AP.Sweep.Reprocess
      .Mode = 3   'Set Measurement Mode to Noise
      AP.Sweep.Reprocess
   End With
```

End Sub

Comment In this macro Fasttrig uses a multitone signal to display the waveform spectrum, system response, distortion, and noise.

AP.S1DSP.FastTrig.FreqRes

Property

Syntax `AP.S1DSP.FastTrig.FreqRes(unit$)`

Data Type Double Valid amplitude settings are from +/- 0.0 to 13.0 %.

Parameters

Name	Description
unit$	String that designates the desired unit. The following unit is valid for this command: %

Description This command sets the Triggered Multitone Tester 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 modes.

In Response mode, 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 mode 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 mode, 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 mode 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 macro `AP.S1DSP.FastTrig.FreqCorrection.`

AP.S1DSP.FastTrig.InputFormat

Property

Syntax `AP.S1DSP.FastTrig.InputFormat`

Data Type Integer

0	A/D
1	Digital.

Description This command sets the Triggered Multitone Tester Input Format.

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.Mode

Property

Syntax `AP.S1DSP.FastTrig.Mode`

Data Type Integer

0	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.
1	Response: is always used with a sweep table (.ADS file) which lists the exact frequencies of the sinewaves in the multitone signal which are to be used for frequency response measurements. The DSP sends to the computer to be plotted only the amplitudes of the FFT bins containing those exact frequencies, resulting in a frequency response graph. There are typically from 3 to 30 sinewaves in most multitone signals.

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 tones 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.

2 Distortion: this selection 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 then 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 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 mode (see below).

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 tones 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.

3 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 FASTTRIG Transform length being set to the value twice the length of the waveform file which generates the multitone signal. The analyzer frequency resolution is twice the resolution of the generated signal. The result is that every alternate analyzer FFT bin falls between bins 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.

Description This command sets the Triggered Multitone Tester measurement mode. The `AP.S1DSP.FastTrig.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 macro `AP.S1DSP.FastTrig.FreqCorrection.`

AP.S1DSP.FastTrig.Output

Property

Syntax `AP.S1DSP.FastTrig.Output`

Data Type Boolean

True	On
False	Off

Description This command sets the Triggered Multitone Tester channel A and B outputs to ON or OFF if they have been individually enabled by the `AP.S1DSP.FastTrig.ChAOutput` and `AP.S1DSP.FastTrig.ChBOutput` commands.

See Also `AP.S1DSP.FastTrig.ChAOutput`, `AP.S1DSP.FastTrig.ChBOutput`

Example See example macro `AP.S1DSP.FastTrig.Ampl`.

AP.S1DSP.FastTrig.TrigCriteria

Property

Syntax `AP.S1DSP.FastTrig.TrigCriteria`

Data Type Integer

0	Normal.
1	Tight.
2	Loose.

Description This command sets the Triggered Multitone Tester Trigger Criteria.

The Triggering Criteria field allows the user to select Tight, Normal, and Loose conditions. Each selection 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 FASTTRIG will not otherwise trigger on highly distorted or noisy signals or signals passed through narrow-band or otherwise non-flat devices.

Example See example macro `AP.S1DSP.FastTrig.FreqCorrection`.

AP.S1DSP.FastTrig.TrigSource

Property

Syntax `AP.S1DSP.FastTrig.TrigSource`

Data Type Integer

0	Auto + 0 ms
1	Auto +100 ms
2	Auto + 200 ms
3	Auto + 400 ms
4	Auto + 1 sec
5	Auto + 2 sec Auto selections all will cause triggering (after the stated time delay) if either channel 1 or channel 2 has a signal amplitude greater than digital zero. These are the normal operating modes of FASTTRIG, with the zero delay selection requiring the shortest signal duration.
6	Dgen: this selection modes only on Dual Domain units (SYS-300 series). If FASTTRIG is generating a signal from a waveform file, a Digital Generator trigger is issued each time the first sample from the file is generated.
7	External: this source is operational only with SYS-322 (Dual Domain) units. It 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 acquisiton being triggered on the next sample after pin 3 is pulled high.
8	Free Run: when selected, signal acquisition begins immediately after F9, `AP.Sweep.Start`, or Go is initiated, regardless of signal amplitude.

Description This command sets the Triggered Multitone Tester Trigger Source & Delay.

In-service multitone testing of broadcast transmission circuits and other similar applications involves inserting a brief burst of multitone signal into a short pause in program material. Broadcast transmission circuits typically include modulation processors and other forms of compressors and limiters. It may be desirable to allow a certain stabilization time following the start of the multitone signal before the measurement is made, in order that the measured conditions represent something approaching steady-state rather than initial transient conditions. The `AP.S1DSP.FastTrig.TrigSource` command permits control of this stabilization time. The duration of multitone signal transmitted must also be lengthened when non-zero values of delay are used.

Example See example macro `AP.S1DSP.FastTrig.FreqCorrection.`

AP.S1DSP.FastTrig.Warnings

ⓘ Property

Syntax **`AP.S1DSP.FastTrig.Warnings`**

Data Type Boolean

`True`	Warnings ON.
`False`	Warnings OFF.

Description This command enables or disables Frequency Error Correction Warnings on Digital Generator waveform data.

When a Digital Generator waveform file is first loaded, FASTTRIG analyzes the multitone signal described in the waveform file and determines whether it contains enough sinewaves across the spectrum above 200 Hz, at sufficient frequency spacing, to provide reliable frequency error correction on an acquired signal. If the `AP.S1DSP.FastTrig.Warnings` command is set to True (ON), a warning message will be displayed whenever a waveform file is loaded which does not meet the built-in criteria. In many cases, the built-in criteria are somewhat conservative and signals with fewer tones or closer spacing may still work properly. Since display of a warning

message will interrupt the flow of the program. Setting the `AP.S1DSP.FastTrig.Warnings` command to False (OFF) provides a means of ignoring the message and avoiding hang-up of the program when it has been experimentally determined that frequency error correction works properly with a particular signal.

See Also `AP.S1DSP.FastTrig.FreqCorrection`

Example See example macro `AP.S1DSP.FastTrig.FreqCorrection.`

AP.S1DSP.FastTrig.WfmName

❶❷ **Method**

Syntax **`AP.S1DSP.FastTrig.WfmName(`*`filename$`*`)`**

Data Type Boolean

Parameters

Name	Description
`filename$`	Any valid DOS path and file name. The file must be an APWIN waveform file (.agm or .ags).

Description This command loads the designated waveform file (.AGM or .AGS) into the Multitone Generator/Analyzer Digital Generator.

Example See example macro `AP.S1DSP.FastTrig.FreqCorrection.`

AP.S1DSP.FastTrig.Window

❶ **Property**

Syntax **`AP.S1DSP.FastTrig.Window`**

Data Type Integer

`0`	None
`1`	Hann

Description This command sets the Triggered Multitone Tester Window selection. See Appendix E for FFT Window Discriptions.

Example See example macro `AP.S1DSP.FastTrig.FreqCorrection.`

User Notes

User Notes

User Notes

Spectrum Analyzer/Generator

AP.S1DSP.FFTGen.Ampl

Property

Syntax **AP.S1DSP.FFTGen.Ampl(*unit$*)**

Data Type Double — Valid amplitude settings are from 0 to 100 %FS.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits

Description This command sets the Spectrum Analyzer/Generator Digital Generator Amplitude.

Example

```
Sub Main
'This procedure produces the graph of an FFT of a 2kHz
sinewave.
   AP.File.OpenTest "FFTGEN3.AT1"
   With AP.S1Dsp.FFTGen
      .WfmName = "Sine.agm"    'Load waveform
      .Ampl("FFS") = .95       'Set DGEN Ampl
      .Freq("Hz") = 2000       'Set DGEN Freq to 2kHz
      .ChAOutput = True        'Set DGEN Ch A on
      .ChBOutput = True        'Set DGEN Ch B on
      .Output = True           'Set DGEN Output on
      .InputFormat = 0         'Set Input to A/D
      .Ch1Source = 0           'Set Ch 1 Source Anlr-A
      .Ch2Source = 0           'Set Ch 2 Source Anlr-A
      .FFTLength = 0           'Set FFT Length to Maximum
      .Averages = 1            'Set num of Averages to 4
      .Window = 0    'Set FFT Window to Blackman-Harris
      .WfmDisplay = 0   'Set Wave Display to Interpolate
      .SubtractAve = True  'Set Subtract Avg Value
      .TrigSource = 4      'Set Trigger Source to DGEN
   End With
   AP.Sweep.Start
End Sub
```

AP.S1DSP.FFTGen.Averages

Property

Syntax `AP.S1DSP.FFTGen.Averages`

Data Type Integer

0	1
1	4
2	16
3	64
4	256
5	1024

Description This command sets the Spectrum Analyzer/Generator number of averages.

FFTGEN 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 range of initial acquisitions. Coherent signals, however, are the same at each acquisition and are not affected by averaging. 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.

Example See example for `AP.S1DSP.FFTGen.Ampl`.

AP.S1DSP.FFTGen.Ch1Rdg

Property

Syntax `AP.S1DSP.FFTGen.Ch1Rdg(unit$)`

Data Type Variant

Parameters

Part	Description

unit$ 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 Spectrum Analyzer/Generator channel 1 Peak Monitor meter and zeros the ready count.

See Also AP.S1DSP.FFTGen.Ch1Ready, AP.S1DSP.FFTGen.Ch1Trig

Example

```
Sub Main
   AP.File.OpenTest "FFTGEN1.AT1"
   With AP.S1Dsp.FFTGen
      .Ampl("FFS") = .95       'Set DGEN Ampl.
      .Freq("Hz") = 2000       'Set DGEN Freq to 2kHz.
      .ChAOutput = True        'Set DGEN Ch A on.
      .Output = True           'Set DGEN Output on.
      .InputFormat = 0         'Set Input to A/D.
      .Ch1Source = 0           'Set Source 1 to Anlr-A.
      Wait 1
      .Ch1Trig                 'Trigger new reading.
      Do
         Ready = .Ch1Ready
         Loop Until Ready > 0 'Wait for new reading.
      Reading = .Ch1Rdg("dBFS")  'Get new reading.
   End With
   NewLine$ = Chr(13)
   a$= "Ch A Peak Mon Reading " _
      & Left(Str$(Reading),6) & "dBFS"

   AP.Prompt.Text = a$ & NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```


AP.S1DSP.FFTGen.Ch1Ready

Property

Syntax AP.S1DSP.FFTGen.Ch1Ready

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Spectrum Analyzer/Generator channel 1 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 `AP.S1DSP.FFTGen.Ch1Rdg` or `AP.S1DSP.FFTGen.Ch1Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.FFTGen.Ch1Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.FFTGen.Ch1Rdg, AP.S1DSP.FFTGen.Ch1Trig`

Example See example for `AP.S1DSP.FFTGen.Ch1Rdg`.

AP.S1DSP.FFTGen.Ch1Source

Property

Syntax **`AP.S1DSP.FFTGen.Ch1Source`**

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.FFTGen.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Gen-Mon
5	DSP A
6	DSP B
7	None

The following list contains the selections relevant to the AP.S1DSP.FFTGen.InputFormat command Digital input selection.

0	A
1	B
2	None

Description This command sets the Triggered Multitone Tester Channnel 1 Input.

Example See example for AP.S1DSP.FFTGen.Ch1Rdg.

AP.S1DSP.FFTGen.Ch1Trig

Method

Syntax **AP.S1DSP.FFTGen.Ch1Trig**

Description Causes a restart of the reading cycle and zeros the ready count for the AP.S1DSP.FFTGen.Ch1Rdg command. The reading in progress is aborted.

See Also AP.S1DSP.FFTGen.Ch1Rdg, AP.S1DSP.FFTGen.Ch1Ready

Example See example for AP.S1DSP.FFTGen.Ch1Rdg.

AP.S1DSP.FFTGen.Ch2Rdg

Property

Syntax **AP.S1DSP.FFTGen.Ch2Rdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	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 Spectrum Analyzer/Generator channel 2 Peak Monitor meter and zeros the ready count.

See Also

`AP.S1DSP.FFTGen.Ch2Ready, AP.S1DSP.FFTGen.Ch2Trig`

Example

```
Sub Main
   AP.File.OpenTest “FFTGEN2.AT1"
   With AP.S1Dsp.FFTGen
      .Ampl(“FFS”) = .95       ‘Set DGEN Ampl
      .Freq(“Hz”) = 2000       ‘Set DGEN Freq to 2kHz
      .ChBOutput = True        ‘Set DGEN Ch B on
      .Output = True           ‘Set DGEN Output on
      .InputFormat = 0         ‘Set Input to A/D
      .Ch2Source = 0           ‘Set Source 2 to Anlr-A
      Wait 1
      .Ch2Trig                 ‘Trigger new reading
      Do
         Ready = .Ch2Ready
         Loop Until Ready > 0 ‘Wait for new reading
      Reading = .Ch2Rdg(“dBFS”)  ‘Get new reading
   End With
   NewLine$ = Chr(13)
   a$= “Ch B Peak Mon Reading ” _
      & Left(Str$(Reading),6) & “dBFS”
   AP.Prompt.Text = a$ & NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.FFTGen.Ch2Ready

Property

Syntax `AP.S1DSPCh2Ready`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Spectrum Analyzer/Generator channel 2 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 `AP.S1DSP.FFTGen.Ch2Rdg` or `AP.S1DSP.FFTGen.Ch2Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.FFTGen.Ch2Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.FFTGen.Ch2Rdg, AP.S1DSP.FFTGen.Ch2Trig`

Example See example for `AP.S1DSP.FFTGen.Ch2Rdg`.

AP.S1DSP.FFTGen.Ch2Source

Property

Syntax `AP.S1DSP.FFTGen.Ch1Source`

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.FFTGen.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Gen-Mon
5	DSP A

6	DSP B
7	None

The following list contains the selections relevant to the `AP.S1DSP.FFTGen.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Triggered Multitone Tester Channnel 2 Input.

Example See example for `AP.S1DSP.FFTGen.Ch2Rdg`.

AP.S1DSP.FFTGen.Ch2Trig

Method

Syntax **`AP.S1DSP.FFTGen.Ch2Trig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.FFTGen.Ch2Rdg` comand. The reading in progress is aborted.

See Also `AP.S1DSP.FFTGen.Ch2Rdg, AP.S1DSP.FFTGen.Ch2Ready`

Example See example for `AP.S1DSP.FFTGen.Ch2Rdg`.

AP.S1DSP.FFTGen.ChAOutput

Property

Syntax **`AP.S1DSP.FFTGen.ChAOutput`**

Data Type Boolean

True	ON.
False	OFF.

Description This command sets Spectrum Analyzer/Generator Generator Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.FFTGen.ChBOutput`

Example See example for `AP.S1DSP.FFTGen.Ch1Rdg`.

AP.S1DSP.FFTGen.ChBOutput

Property

Syntax **`AP.S1DSP.FFTGen.ChBOutput`**

Data Type Boolean

`True`	ON.
`False`	OFF.

Description This command sets the Spectrum Analyzer/Generator Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.FFTGen.ChAOutput`

Example See example for `AP.S1DSP.FFTGen.Ch2Rdg`.

AP.S1DSP.FFTGen.FFTLength

Property

Syntax **`AP.S1DSP.FFTGen.FFTLength`**

Data Type Integer

`0`	Maximum
`1`	4096
`2`	2048
`3`	1024
`4`	512
`5`	256

Description This command sets the Spectrum Analyzer/Generator FFT Length.

Maximum means 16384 on all System One Dual Domain units and on System One Plus DSP units with the MEM (memory) option, but is the same as the 4096 selection on System One Plus DSP units without the MEM option.

The `AP.S1DSP.FFTGen.FFTLength` command controls the record length (number of words of memory, in samples) which will be filled when the `AP.Sweep.Start` (F9/Go) command is executed. The FFT Length field value also controls the record length used as input to the FFT process when either `AP.Sweep.Start` (F9/Go) is initiated to acquire and transform, or the `AP.Sweep.Reprocess` (F6 function key) is used to re-transform 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 See example for `AP.S1DSP.FFTGen.Ampl`.

AP.S1DSP.FFTGen.Freq

Property

Syntax **`AP.S1DSP.FFTGen.Freq(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
`unit$`	String that designates the desired unit. The following unit is valid for this command: Hz

Description This command sets the Spectrum Analyzer/Generator Digital Generator Frequency.

Example See example for `AP.S1DSP.FFTGen.Ampl`.

AP.S1DSP.FFTGen.InputFormat

Property

Syntax **`AP.S1DSP.FFTGen.InputFormat`**

Data Type Integer

`0`	A/D

1	Digital

Description This command sets the Spectrum Analyzer/Generator Input Format.

Example See example for `AP.S1DSP.FFTGen.Ampl`.

AP.S1DSP.FFTGen.Output

Property

Syntax **`AP.S1DSP.FFTGen.Output`**

Data Type Boolean

True	On
False	Off

Description This command sets the Spectrum Analyzer/Generator channel A and B outputs to ON or OFF if they have been individually enabled by the `AP.S1DSP.FFTGen.ChAOutput` and `AP.S1DSP.FFTGen.ChBOutput` commands.

See Also `AP.S1DSP.FFTGen.ChAOutput`, `AP.S1DSP.FFTGen.ChBOutput`

Example See example for `AP.S1DSP.FFTGen.Ampl`.

AP.S1DSP.FFTGen.SubtractAve

Property

Syntax **`AP.S1DSP.FFTGen.SubtractAve`**

Data Type Boolean

True	Subtract average value ON.
False	Subtract average value OFF.

Description This command enables or disables computation of the average value of all samples in the acquisition buffer and the subtraction of that computed value from the value of each sample before an FFT transform or processing the values according to the Wave Display field. The effect of the Subtract Average Value function is very similar to using 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 scale.

Example See example for `AP.S1DSP.FFTGen.Ampl`.

AP.S1DSP.FFTGen.TrigSource ❶ Property

Syntax `AP.S1DSP.FFTGen.TrigSource`

Data Type Integer

0	Free Run: signal acquisition begins immediately after F9, `AP.Sweep.Start,` or Go is initiated, regardless of signal amplitude.
1	Channel #1: will cause a triggering when the Analog Analyzer channel A input following input ranging has a positive-going zero crossing signal with an amplitude greater than 0.1%FS.
2	Channel #2: will cause a triggering when the Analog Analyzer channel B input following input ranging has a positive-going zero crossing signal with an amplitude greater than 0.1%FS.
3	Auto: will cause triggering if either channel 1 or channel 2 has a signal amplitude greater than digital zero.
4	Dgen: this selection functions only on Dual Domain units (SYS-300 series). If FASTTRIG is generating a signal from a waveform file, a Digital Generator trigger is issued each time the first sample from the file is generated.

Description This command sets the Spectrum Analyzer/Generator Trigger Source & Delay.

Example See example for `AP.S1DSP.FFTGen.Ampl`.

AP.S1DSP.FFTGen.Wfm ❶❷ Property

Syntax `AP.S1DSP.FFTGen.Wfm`

Data Type Integer

0	Sine
1	Arb

Description This command selects the Spectrum Analyzer/Generator waveform.

Example

```
Sub Main
'This procedure produces an FFT of a 1kHz sinewave.
   AP.Application.NewTest
   AP.Gen.Wfm 17
   AP.Gen.Output = True
   AP.Anlr.ChAInput = 1
   AP.S1Dsp.Program = 2    'Spectrum analyzer/generator
   With AP.S1Dsp.FFTGen
      .Wfm = 0 'Sine Waveform
      .Ampl("FFS") = 1.0   'Set DGEN Ampl
      .Freq("Hz") = 1000   'Set DGEN Freq to 1kHz
      .ChAOutput = True    'Set DGEN Ch A on
      .Output = True       'Set DGEN Output on
      .InputFormat = 0     'Set Input to A/D
      .Ch1Source = 0       'Set Ch 1 Source Anlr-A
      .Ch2Source = 7       'Set Ch 2 Source None
      .FFTLength = 0       'Set FFT Length to Maximum
      .Window = 0 'Set FFT Window to Blackman-Harris
      .TrigSource = 4      'Set Trigger Source to DGEN
   End With
   AP.Sweep.Source1.Id = 5515
   AP.Sweep.Data1.Id = 6023
   AP.Sweep.Data1.Top("dBV") = 0.000000
   AP.Sweep.Start
   AP.Graph.OptimizeLeft
End Sub
```


AP.S1DSP.FFTGen.WfmDisplay

Property

Syntax `AP.S1DSP.FFTGen.WfmDisplay`

Data Type Integer

0	Interpolate
1	Display Samples

2	Peak Values
3	Absolute Values

Description

This command sets the Spectrum Analyzer/Generator 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 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 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 thus always sends a positive number to the computer. The advantage of Absolute Values mode is that logarithms may be computed when all numbers involved are positive, so a dB units may be used on the Y axis to display the waveform. Waveform display with Absolute Values mode thus can create a wide dynamic range oscilloscope which displays the envelope of an audio signal, calibrated in familiar dB units such as dBV, dBm, 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 `AP.S1DSP.FFTGen.Ampl`.

AP.S1DSP.FFTGen.WfmName ❶❷ Method

Syntax **`AP.S1DSP.FFTGen.WfmName(`*`filename$`*`)`**

Data Type Boolean

Parameters

Name	Description
`filename$`	Any valid DOS path and file name. The file must be an APWIN waveform file (.agm or .ags).

Description This command loads the designated waveform file (.AGM or .AGS) into the Spectrum Analyzer/Generator Digital Generator.

Example See example for `AP.S1DSP.FFTGen.Ampl`.

AP.S1DSP.FFTGen.Window

Property

Syntax `AP.S1DSP.FFTGen.Window`

Data Type Integer

0	Blackman-Harris
1	Hann
2	Flat-Top
3	None

Description This command sets the Spectrum Analyzer/Generator Window selection. See Appendix E for FFT Window Discriptions.

Example See example for `AP.S1DSP.FFTGen.Ampl`.

User Notes

User Notes

User Notes

User Notes

AP.S1DSP.FFTSlide.Ch1Rdg

Property

Syntax **AP.S1DSP.FFTSlide.Ch1Rdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	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 Spectrum Analyzer channel 1 Peak Monitor meter and zeros the ready count.

See Also AP.S1DSP.FFTSlide.Ch1Ready, AP.S1DSP.FFTSlide.Ch1Trig

Example

```
Sub Main
   AP.File.OpenTest “SLIDE1.AT1" ‘Open test
   With AP.S1Dsp.FFTSlide
      .InputFormat = 0          ‘Set Input to A/D
      .Ch1Source = 0            ‘Set Source 1 to Anlr-A
      Wait 1
      .Ch1Trig                  ‘Trigger a new reaading
      Do
         Ready = .Ch1Ready
         Loop Until Ready > 0 ‘Wait for new reading
      Reading1 = .Ch1Rdg(“dBFS”) ‘Get new reading
   End With
   NewLine$ = Chr(13)
   a$= “Ch 1 Peak Mon  ” & Left(Str$(Reading1),6) _
      & “dBFS”
   AP.Prompt.Text = a$ & NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.FFTSlide.Ch1Ready

Property

Syntax `AP.S1DSP.FFTSlide.Ch1Ready`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Spectrum Analyzer channel 1 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 `AP.S1DSP.FFTSlide.Ch1Rdg` or `AP.S1DSP.FFTSlide.Ch1Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.FFTSlide.Ch1Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.FFTSlide.Ch1Rdg`, `AP.S1DSP.FFTSlide.Ch1Trig`

Example See example for `AP.S1DSP.FFTSlide.Ch1Rdg`.

AP.S1DSP.FFTSlide.Ch1Source

Property

Syntax `AP.S1DSP.FFTSlide.Ch1Source`

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.FFTSlide.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl

3	Anlr Reading Ratio
4	Gen-Mon
5	DSP A
6	DSP B
7	None

The following list contains the selections relevant to the `AP.S1DSP.FFTSlide.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Spectrum Analyzer Channnel 1 Input.

Example See example for `AP.S1DSP.FFTSlide.Ch1Rdg`.

AP.S1DSP.FFTSlide.Ch1Trig

Method

Syntax **`AP.S1DSP.FFTSlide.Ch1Trig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.FFTSlide.Ch1Rdg` command. The reading in progress is aborted.

See Also `AP.S1DSP.FFTSlide.Ch1Rdg`, `AP.S1DSP.FFTSlide.Ch1Ready`

Example See example for `AP.S1DSP.FFTSlide.Ch1Rdg`.

AP.S1DSP.FFTSlide.Ch2Rdg

Property

Syntax **`AP.S1DSP.FFTSlide.Ch2Rdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description
unit$	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 Spectrum Analyzer channel 2 Peak Monitor meter and zeros the ready count.

See Also AP.S1DSP.FFTSlide.Ch2Ready, AP.S1DSP.FFTSlide.Ch2Trig

Example

```
Sub Main
   AP.File.OpenTest "SLIDE1.AT1" 'Open test
   With AP.S1Dsp.FFTSlide
      .InputFormat = 0          'Set Input to A/D
      .Ch2Source = 0            'Set Source 2 To Anlr-A
      Wait 1
      .Ch2Trig                  'Trigger a new Reading
      Do
         Ready = .Ch2Ready
      Loop Until Ready > 0      'Wait For new reading
      Reading1 = .Ch2Rdg("dBFS") 'Get new reading
   End With
   NewLine$ = Chr(13)
   a$= "Ch2 Peak Mon  " & Left(Str$(Reading1),6) _
      & "dBFS"
   AP.Prompt.Text = a$ & NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.FFTSlide.Ch2Ready

Property

Syntax **AP.S1DSP.FFTSlide.Ch2Ready**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Spectrum Analyzer channel 2 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 `AP.S1DSP.FFTSlide.Ch2Rdg` or `AP.S1DSP.FFTSlide.Ch2Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.FFTSlide.Ch2Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.FFTSlide.Ch2Rdg`, `AP.S1DSP.FFTSlide.Ch2Trig`

Example See example for `AP.S1DSP.FFTSlide.Ch2Rdg`.

AP.S1DSP.FFTSlide.Ch2Source

Property

Syntax **`AP.S1DSP.FFTSlide.Ch1Source`**

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.FFTSlide.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Gen-Mon
5	DSP A
6	DSP B
7	None

The following list contains the selections relevant to the `AP.S1DSP.FFTSlide.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Spectrum Analyzer Channnel 2 Input.

Example See example for `AP.S1DSP.FFTSlide.Ch2Rdg`.

AP.S1DSP.FFTSlide.Ch2Trig

Method

Syntax **`AP.S1DSP.FFTSlide.Ch2Trig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.FFTSlide.Ch2Rdg` comand. The reading in progress is aborted.

See Also `AP.S1DSP.FFTSlide.Ch2Rdg`, `AP.S1DSP.FFTSlide.Ch2Ready`

Example See example for `AP.S1DSP.FFTSlide.Ch2Rdg`.

AP.S1DSP.FFTSlide.FFTLength

Property

Syntax **`AP.S1DSP.FFTSlide.FFTLength`**

Data Type Integer

0	Maximum
1	4096
2	2048
3	1024
4	512

5	256

Description

This command sets the Spectrum Analyzer FFT Length.

Maximum means 16384 on all System One Dual Domain units and on System One Plus DSP units with the MEM (memory) option, but is the same as the 4096 selection on System One Plus DSP units without the MEM option.

The `AP.S1DSP.FFTSlide.FFTLength` command controls the record length (number of words of memory, in samples) which will be filled when the `AP.Sweep.Start` (F9/Go) command is executed. The FFT Length field value also controls the record length used as input to the FFT process when either `AP.Sweep.Start` (F9/Go) is initiated to acquire and transform, or the `AP.Sweep.Reprocess` (F6 function key) is used to re-transform a record previously acquired. Longer transform lengths produce greater frequency resolution in the resulting FFT, but require longer times to acquire and to transform the signal.

Example

```
Sub Main
   AP.File.OpenTest "SLIDE2.AT1"    'Open test
   With AP.S1Dsp.FFTSlide
      .FFTLength = 5    'Set FFT Length to 256
      .Window = 2       'Set window to Flat-Top
      .TrigPolarity = 1 'Set Trigger Polarity Positive
      .TrigSource = 5   'Set Trigger Source to Gen Sync
      .StartTime("sec") = 0   'Set FFT Start Time to 0mS
      AP.Sweep.Start
      .StartTime("sec") = .075   'Set FFT Start to 75mS
   End With
   AP.Sweep.Retransform
End Sub
```

AP.S1DSP.FFTSlide.InputFormat

Property

Syntax `AP.S1DSP.FFTSlide.InputFormat`

Data Type Integer

0	A/D
1	Digital.

Description This command sets the Spectrum Analyzer Input Format.

Example See example for `AP.S1DSP.FFTSlide.Ch1Rdg`.

AP.S1DSP.FFTSlide.PreTrig

Property

Syntax **`AP.S1DSP.FFTSlide.PreTrig(`***`unit$`***`)`**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: sec.

Description This command sets the Spectrum Analyzer Pre-Trigger time.

FFTSLIDE has the ability to fill the acquisition buffer with signal samples starting at a user-defined time before the trigger occurs and 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 Pre-Trigger Time field determines how much time (and thus how many samples) prior to the trigger event are retained. The total length of signal acquired will be as described 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 640 milliseconds. If the Pre-Trigger Time value is -50 milliseconds, for example, then 590 additional milliseconds of signal following the trigger will also be acquired to fill the entire 640 ms buffer.

Pre-trigger data is acquired in this fashion: when the F9 key is pressed or Go is clicked, FFTSLIDE 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, FFTSLIDE 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

```
Sub Main
   AP.File.OpenTest "SLIDE3.AT1" 'Open test
   With AP.S1Dsp.FFTSlide
      .WfmDisplay = 0   'Set Wave Display to Interpolate
      .SubtractAve = True  'Set Subtract Average Value
      .PreTrig("sec") = 0  'Set Pre-trigger Time to 0
      AP.Sweep.Start
      .PreTrig("sec") = -.01 'Set Pre-trigger to -10mS
   End With
   AP.Gen.BurstOnTime("Cycles") = 1
   AP.Gen.Freq("Hz") = 2000
   AP.Sweep.Start
End Sub
```

AP.S1DSP.FFTSlide.StartTime

Property

Syntax **AP.S1DSP.FFTSlide.StartTime(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: sec.

Description This command sets the Spectrum Analyzer Start Time.

Unlike the other System One FFT programs which always compute the transform on the portion of the record beginning with the first sample, FFTSLIDE 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 FFT Length field.

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 `AP.S1DSP.FFTSlide.FFTLength`.

AP.S1DSP.FFTSlide.SubtractAve

Property

Syntax **`AP.S1DSP.FFTSlide.SubtractAve`**

Data Type Boolean

`True`	Subtract average value ON.
`False`	Subtract average value OFF.

Description This command enables or disables computation of the average value of all samples in the acquisition buffer and subtracts 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 very similar to having used AC coupling before acquiring the signal, as long as no signal peaks exceed 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 cause the signal to be off-screen at the selected vertical span.

Example See example for `AP.S1DSP.FFTSlide.PreTrig`.

AP.S1DSP.FFTSlide.TrigPolarity

Property

Syntax **`AP.S1DSP.FFTSlide.TrigPolarity`**

Data Type Integer

`0`	OFF: is free running and acquires immediately after the F9 key is pressed, without waiting for any trigger event.

1	Positive: time zero will be the first positive-going zero crossing of the trigger signal selected in the Trigger Source field.
2	Negative: time zero will be the first negative-going zero crossing of the selected trigger signal.

Description This command sets the Spectrum Analyzer Trigger Polarity.

The flexible hardware triggering capability of FFTSLIDE permits triggering upon either a positive-going or negative-going signal transition. The Trigger Polarity field (visible only on the large version of the Digital Analyzer panel) permits selection of the desired polarity.

Example See example for `AP.S1DSP.FFTSlide.FFTLength`.

AP.S1DSP.FFTSlide.TrigSource

Property

Syntax `AP.S1DSP.FFTSlide.TrigSource`

Data Type Integer

0	Ch. A Input: will cause triggering when the Analog Analyzer channel A input, following input ranging, has a positive-going zero crossing signal with an amplitude greater than 0.1%FS.
1	Ch. B Input: will cause triggering when the Analog Analyzer channel B input, following input ranging, has a positive-going zero crossing signal with an amplitude greater than 0.1%FS.
2	DSP A: the DSP input "A" BNC connector, located on the front panel of SYS-222 and early SYS-322 units and on the rear of SYS-322 units which have optical connectors on the front panel.
3	DSP B: the DSP input "B" BNC connector, located on the front panel of SYS-222 and early SYS-322 units and on the rear of SYS-322 units which have optical connectors on the front panel.
4	Reading Signal: the output of the Analog Analyzer reading meter, following all filtering.
5	Generator Sync: the Sync Output BNC on the Generator Auxiliary Signals panel, on the rear of all DSP units. 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.

6 AC Mains: Power line frequency.

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 involved are positive, so a dB units may be used on the Y axis to display the waveform. Waveform display with Absolute Values mode thus can create a wide dynamic range oscilloscope which displays the envelope of an audio signal, calibrated in familiar dB units such as dBV, dBm, 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 `AP.S1DSP.FFTSlide.PreTrig`.

AP.S1DSP.FFTSlide.WfmDisplay

Property

Syntax `AP.S1DSP.FFTSlide.WfmDisplay`

Data Type Integer

0	Interpolate
1	Display Samples
2	Peak Values
3	Absolute Values

Description This command sets the 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, each 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 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 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 thus always sends a positive number to the computer. The advantage of Absolute Values mode is that logarithms may be computed when all numbers involved are positive, so a dB units may be used on the Y axis to display the waveform. Waveform display with Absolute Values mode thus can create a wide dynamic range oscilloscope which displays the envelope of an audio signal, calibrated in familiar dB units such as dBV, dBm, 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 `AP.S1DSP.FFTSlide.PreTrig`.

AP.S1DSP.FFTSlide.Window

Property

Syntax `AP.S1DSP.FFTSlide.Window`

Data Type Integer

0	Blackman-Harris
1	Hann
2	Flat-Top
3	None

Description This command sets the Spectrum Analyzer Window selection. See Appendix E for FFT Window Discriptions.

Example See example for `AP.S1DSP.FFTSlide.FFTLength`.

User Notes

User Notes

User Notes

User Notes

User Notes

Digital Domain Audio Tester

AP.S1DSP.GenAnlr.Ampl

Property

Syntax **AP.S1DSP.GenAnlr.Ampl(*unit$*)**

Data Type Double — Valid amplitude settings are from 0 to 100 %FS.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits.

Description This command sets the Digital Domain Audio Tester Digital Generator Amplitude.

Example

```
Sub Main
   AP.File.OpenTest "GENANLR1.AT1"
   With AP.S1Dsp.GenAnlr
      .Ampl("FFS") = .95          'Set DGEN Ampl.
      .Freq("Hz") = 1500          'Set DGEN Freq.
      .ChAOutput = True           'Set Ch A on
      .ChBOutput = True           'Set Ch B on
      .Output = True              'Set DGEN on
      .FuncInput = 1        'Set Function meter to Ch B
      .FuncMode = 0         'Set 2 Channel mode
      .FreqSettling( .5, .2, "Hz", 3, .03, 1)
      .FreqTrig             'Trigger new Freq. reading
      Do
      Ready1 = .FreqReady
      Loop Until Ready1 > 0   'Wait for Freq. reading
      Reading1 = .FreqRdg("Hz")  'Get Freq. reading
      .LevelSettling( 1, .00001, "FFS", 3, .1, 1)
      .LevelTrig            'Trigger new Level reading
      Do
      Ready2 = .LevelReady
      Loop Until Ready2 > 0   'Wait for Levelreading
      Reading2 = .LevelRdg("FFS")   'Get Level reading
   End With
   NewLine$ = Chr(13)
   a$= "Ch A Freq " & Left(Str$(Reading1),6) & "Hz"
```

```
    b$= "Ch A Level Mon " & Left(Str$(Reading2),6) & "FFS"
    AP.Prompt.Text = a$ & NewLine$ & b$ & NewLine
    AP.Prompt.ShowWithContinue
    Beep
    Stop
End Sub
```

AP.S1DSP.GenAnlr.ChAOutput

Property

Syntax **AP.S1DSP.GenAnlr.ChAOutput**

Data Type Boolean

True	ON.
False	OFF.

Description This command sets Digital Domain Audio Tester Generator Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also AP.S1DSP.GenAnlr.ChBOutput

Example See example for AP.S1DSP.GenAnlr.Ampl.

AP.S1DSP.GenAnlr.ChBOutput

Property

Syntax **AP.S1DSP.GenAnlr.ChBOutput**

Data Type Boolean

True	ON.
False	OFF.

Description This command sets the Digital Domain Audio Tester Generator Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.GenAnlr.ChAOutput`

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.EqAmpl ❷ Property

Syntax **`AP.S1DSP.GenAnlr.EqAmpl(`*`unit$`*`)`**

Data Type Double — Valid amplitude settings are 0.0 to 100 %FS.

Parameters

Name	Description
`unit$`	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits

Description This command sets the Digital Domain Audio Tester channel A and B post Eq amplitude.

See Also `AP.S1DSP.GenAnlr.EqCurve`

Example

```
Sub Main
   AP.Application.NewTest
   AP.Application.PanelClose apbPanelAnalogGenSmall
   AP.Application.PanelClose apbPanelAnlrSmall
   AP.S1Dsp.Program = 1    'Load Digital Domain Audio _
      Tester (GENANLR)
   AP.Application.PanelOpen apbPanelDSPLarge
   AP.S1Dio.InFormat = 2      'Monitor Digital Generator
   If AP.S1Dsp.GenAnlr.EqCurve("75us-de.adq", 1) _
      = False Then
      AP.Prompt.Text = "EQ Curve file load FAILED."
      AP.Prompt.FontSize = 8  'Set font size to 8 point.
      AP.Prompt.Position(-1,-1,190,120) 'Set location _
         and size.
      AP.Prompt.ShowWithContinue    'Display prompt _
         with Continue button.
      Stop                          'Stop macro.
   End If
   AP.S1Dsp.GenAnlr.Freq("Hz") = 100.0
```

```
   AP.S1Dsp.GenAnlr.Wfm = 1   'Select EQ Sine Waveform
   AP.S1Dsp.GenAnlr.EqAmpl("dBFS") = 0.0  'Set _
      Equalized Am[plitude at 100Hz to 0.0dBFS
   AP.S1Dsp.GenAnlr.Output = True   'Turn Output ON
   AP.Sweep.Data1.Id = 6014
   AP.Sweep.Data1.Top("dBFS") = 0.0    'Set Data 1 _
      Vertical Scale
   AP.Sweep.Data1.Bottom("dBFS") = -20.0
   AP.Sweep.Source1.Id = 5540

   AP.Sweep.Stereo = True           'Stereo Sweep
   AP.Sweep.Start
End Sub
```

AP.S1DSP.GenAnlr.EqCurve

Method

Syntax **AP.S1DSP.GenAnlr.EqCurve(***filename$, column%***)**

Data Type Boolean

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN Eq file (.adq).
column%	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 Digital Domain Audio Tester. Values in the file will be used as multiply factors in czlculating the Digital Generator amplitude values.

See Also `AP.S1DSP.GenAnlr.EqAmpl`

Example See example for `AP.S1DSP.GenAnlr.EqAmpl`.

AP.S1DSP.GenAnlr.FilterHP

ⓘ Property

Syntax **AP.S1DSP.GenAnlr.FilterHP**

Data Type Integer

0	< 10 Hz
1	22 Hz
2	100 Hz
3	400 Hz

Description This command selects the Digital Domain Audio Tester High Pass filter for the function meter.

Example

```
Sub Main
   AP.File.OpenTest “GENANLR2.AT1"
   AP.S1DSP.GenAnlr.FuncInput = 0 ‘Set Function Input _
      to Ch A
   AP.S1DSP.GenAnlr.FuncMode = 5 ‘Set Function Mode to _
      Bandpass
   AP.S1DSP.GenAnlr.FuncBPBRTuning = 2    ‘Set BP/BR _
      Tuning to DGEN
   AP.S1DSP.GenAnlr.FuncBPBRFreq(“Hz”) = 2000 ‘Set _
      BP/BR Filter Freq to 2kHz
   AP.S1DSP.GenAnlr.FuncBPHarmonic = 0       ‘Set BP _
      Harmonic to Fundamental
   AP.S1DSP.GenAnlr.FilterHP = 0 ‘Set HP Filter Freq _
      to 22 Hz
   AP.S1DSP.GenAnlr.RdgRate = 2  ‘Set Reading Rate to _
      8/sec RMS
   AP.Sweep.Start
   AP.Compute.Linearity.Data(1) = True
   AP.Compute.Linearity.Start(“dBFS”) = 0
   AP.Compute.Linearity.Stop(“dBFS”) = -120
   AP.Compute.Linearity.Apply
   AP.Sweep.Data1.Top(“dBFS”) = .5
   AP.Sweep.Data1.Bottom(“dBFS”) = -.5
End Sub
```

AP.S1DSP.GenAnlr.Freq

Property

Syntax `AP.S1DSP.GenAnlr.Freq(unit$)`

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following unit is valid for this command: Hz

Description This command sets the Digital Domain Audio Tester Digital Generator frequency.

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.FreqRdg

Property

Syntax `AP.S1DSP.GenAnlr.FreqRdg(unit$)`

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: Hz, F/R, %Hz, cent, octs, decs, d%, PPM.

Description This command returns a settled reading for the Digital Domain Audio Tester Frequency meter and zeros the ready count.

See Also `AP.S1DSP.GenAnlr.FreqReady`, `AP.S1DSP.GenAnlr.FreqSettling`, `AP.S1DSP.GenAnlr.FreqTrig`

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.FreqReady

Property

Syntax `AP.S1DSP.GenAnlr.FreqReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Domain Audio Tester channel A Frequency 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 `AP.S1DSP.GenAnlr.FreqRdg` or `AP.S1DSP.GenAnlr.FreqTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.GenAnlr.FreqRdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.GenAnlr.FreqRd`, `AP.S1DSP.GenAnlr.FreqSettling`, `AP.S1DSP.GenAnlr.FreqTrig`

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.FreqSettling

Method

Syntax **`AP.S1DSP.GenAnlr.FreqSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.S1DSP.GenAnlr.FreqRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.S1DSP.GenAnlr.FreqRdg`, `AP.S1DSP.GenAnlr.FreqReady`, `AP.S1DSP.GenAnlr.FreqTrig`

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.FreqTrig

Method

Syntax `AP.S1DSP.GenAnlr.FreqTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.GenAnlr.FreqRdg` command. The reading in progress is aborted.

See Also `AP.S1DSP.GenAnlr.FreqRdg`, `AP.S1DSP.GenAnlr.FreqReady`, `AP.S1DSP.GenAnlr.FreqSettling`

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.FuncBPBRFreq

Property

Syntax `AP.S1DSP.GenAnlr.FuncBPBRFreq(`*`unit$`*`)`

Data Type Double

Parameters

Name	Description
unit$	The following units are available Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.

Description This command sets the Digital Domain Audio Tester bandpass/bandreject filter to the frequency value passed.

In Bandpass or Crosstalk functions it may be tuned to any center frequency from 0.04% to 40% of the sample rate; for example, from 20 Hz to 19.2 kHz at the 48 kHz rate. The filter is a 10-pole design with a nominal 1/13 octave bandwidth (Q of 19, -3 dB bandwidth of approximately 5.2% of center frequency). The response will become non-symmetrical as the center frequency approaches 20 kHz. In THD+N modes, the filter tunes from 0.1% to 40% of the sample rate; for example, from 50 Hz to 19.2 kHz at the 48 kHz rate. Several modes are available for steering this filter frequency in order to simplify swept measurements.

See Also `AP.S1DSP.GenAnlr.FuncMode`

Example See example for `AP.S1DSP.GenAnlr.FilterHP`.

AP.S1DSP.GenAnlr.FuncBPBRTuning

Property

Syntax `AP.S1DSP.GenAnlr.FuncBPBRTuning`

Data Type Integer

0	Panel: BP/BR filter is fixed-tuned at the frequency entered into the BP/BR Filter Freq field just below the BP/BR Tuning field. The Digital Analyzer panel must be displayed in its large version for the BP/BR Filter Freq field to be visible. This tuning mode is often used when performing linearity tests (wide range amplitude sweeps) of A/D converters, since the digital frequency counter of the GENANLR program will not be operational at extremely low signal levels.
1	DGen: BP/BR filter center frequency is automatically tuned to the frequency of the sinewave generator of GENANLR, whether fixed in panel mode (at the bottom of the large version of the Digital Analyzer panel) or being swept as the Source 1 or Source 2 parameter. This tuning mode provides the most rapid and reliable operation during frequency sweeps of digital input-digital output devices where GENANLR also provides the test signal.
2	Dig Freq: BP/BR filter is tuned to the frequency being measured by the digital domain frequency counter as displayed near the top of the Digital Analyzer panel. This tuning mode is convenient for all types of digital domain measurements as long as the digital signal level is high enough to provide reliable, noise-free operation of the digital domain frequency counter, but will not be quite as fast in settling as the DGen mode.
3	Anlr BP/BR: DSP-implemented digital domain BP/BR filter tracks the frequency of the Analog Analyzer hardware BP/BR filter.
4	Anlg Gen: DSP BP/BR filter trackes the frequency of the Analog Generator. This is useful when measuring A/D converters.

Description This command sets the Digital Domain Audio Tester bandpass/bandreject filter Tuning Source.

Example See example for `AP.S1DSP.GenAnlr.FilterHP`.

AP.S1DSP.GenAnlr.FuncBPHarmonic

Property

Syntax `AP.S1DSP.GenAnlr.FuncBPHarmonic`

Data Type Integer

0	Fundamental
1	2nd Harmonic
2	3nd Harmonic
3	4nd Harmonic
4	5nd Harmonic

Description This command sets the Digital Domain Audio Tester bandpass filter so that it may be automatically tuned to the Fundamental, 2nd, 3rd, 4th, or 5th Harmonic. This permits automatic tracking of the fundamental frequency or a specific harmonic from 2nd through 5th.

Example See example for `AP.S1DSP.GenAnlr.FilterHP`.

AP.S1DSP.GenAnlr.FuncInput

Property

Syntax `AP.S1DSP.GenAnlr.FuncInput`

Data Type Integer

0	Channel A
1	Channel B

Description This command selects the Digital Domain Audio Tester channel A or channel B to be used for measurement with the Function meter.

See Also `AP.S1DSP.GenAnlr.RdgRate`, `AP.S1DSP.GenAnlr.FuncMode`

Example See example for `AP.S1DS`

P.GenAnlr.Ampl.

AP.S1DSP.GenAnlr.FuncMode

Property

Syntax `AP.S1DSP.GenAnlr.FuncMode`

Data Type Integer

Description This command selects the analysis mode of the Digital Domain Audio Tester Function meter.

The measurement is taken from the selected channel, using the selected mode, and using the units specified by that mode.

If a reading is not ready when this command is called, it will wait for a reading to become available. Any particular reading will be returned only once.

0	2-Channel: The function meter connects to the channel selected by the `AP.S1DSP.GenAnlr.FuncInput` command, while the Level monitor connects to the opposite channel. The bandpass/bandreject filter is not used. This function provides simultaneous readings of level on both stereo channels and thus permits frequency response measurements of digital stereo devices in a single sweep.
1	Ratio: The function meter connects to the channel selected by the `AP.S1DSP.GenAnlr.FuncInput` command, while the Level monitor connects to the opposite channel. The bandpass/bandreject filter is not used. The display of the function meter is not the directly-measured amplitude of the selected channel, but is the amplitude ratio (in x/y or % units) or level difference (in dB units) of the two channels. Ratio function thus provides an interchannel balance or differential gain measurement, or a non-frequency-selective crosstalk measurement. Ratio mode works only when the selected channel amplitude is equal to or less than the amplitude on the opposite channel. If the amplitude on the selected channel is greater than the opposite channel, the display will be clipped at 0 dB and it will be necessary to select the other channel to obtain a reading.
2	Crosstalk: The function meter connects to the channel selected by the `AP.S1DSP.GenAnlr.FuncInput` command, while the Level monitor connects to the opposite

channel. The bandpass filter processes the signal before the function meter measures it. The display of the function meter is not the directly-measured amplitude of the selected channel, but is the amplitude ratio (in x/y or % units) or level difference (in dB units) of the two channels. Crosstalk function thus provides a frequency-selective crosstalk measurement, permitting accurate crosstalk measurements even when the crosstalk signal is lower in amplitude than the wideband noise level. Crosstalk mode works only when the selected channel amplitude is equal to or less than the amplitude on the opposite channel. If the amplitude on the selected channel is greater than the opposite channel, the display will be clipped at 0 dB and it will be necessary to select the other channel to obtain a reading.

3 THD+N relative:Tthe function meter and the Level monitor both connect to the channel selected by the A or B radio buttons. The bandreject (notch) filter processes the signal to remove the fundamental signal before the function meter measures it. The display of the function meter is not the directly-measured amplitude of the notch-filtered channel, but is the amplitude ratio (in x/y or % units) or level difference (in dB units) of the function meter with respect to the Level meter, which is measuring the unfiltered input signal. The result is THD+N relative to the input signal.

4 THD+N absolute: The function meter and the Level monitor both connect to the channel selected by the `AP.S1DSP.GenAnlr.FuncInput` command. The bandreject (notch) filter processes the signal to remove the fundamental signal before the function meter measures it. The display of the function meter is the directly-measured amplitude of the notch-filtered channel and thus does not depend upon the reading of the Level monitor. The available units (obtained by clicking on the down arrow at the right of the function meter display field) are the normal digital domain absolute amplitude units of %FS, FFS, dBFS, or bits. Digital domain THD+N in absolute units is particularly useful when performing amplitude sweeps on a digital I/O device or an A/D converter, where use of relative units obscures the basic

noise-limited operation of many devices. An ideal device will exhibit constant THD+N at all amplitudes.

5 Bandpass: the function meter and the Level monitor both connect to the channel selected by the `AP.S1DSP.GenAnlr.FuncInput` command. The bandpass filter processes the signal to remove the fundamental signal before the function meter measures it. The display of the function meter is the directly-measured amplitude of the notch-filtered channel and does not depend upon the reading of the Level monitor. Digital domain bandpass function is useful when measuring very low-amplitude signals, approaching or below the wideband noise level, as is commonly done when making input/output linearity (amplitude) sweeps on a digital I/O device or an A/D converter.

6 Noise - Unweighted: the function meter and the Level monitor both connect to the channel selected by the `AP.S1DSP.GenAnlr.FuncInput` command. Neither the bandpass/bandreject filter nor either of the weighting filters is connected in the function meter circuit, but a high-pass filter may be selected. The display of the function meter is the directly-measured amplitude of the selected channel and does not depend upon the reading of the Level monitor. The Noise unweighted function in the digital domain corresponds to band-limited, unweighted noise measurements in the analog domain. The Noise unweighted function may also be used with the 400 Hz high-pass filter selected and stimulus from a low-frequency signal to make quantization noise and distortion measurements on digital signals.

7 Noise - “A” Weighted: the main meter and the Level monitor both connect to the channel selected by the `AP.S1DSP.GenAnlr.FuncInput` command. The DSP-implemented A-weighting filter is connected in the function meter circuit. The DSP-implemented true RMS detector should be selected, typically at a low reading rate such as 4/second, in the Reading Rate field. The display of the function meter does not depend upon the reading of the Level monitor.

	8	Noise - CCIR Weighted: the function meter and the Level monitor both connect to the channel selected by the `AP.S1DSP.GenAnlr.FuncInput` command. The DSP-implemented CCIR-468 weighting filter is connected in the function meter circuit. For measurements conforming to the CCIR-468 standard, the DSP-implemented quasi-peak detector should be selected by the 4/sec QPK selection in the Reading Rate field. The CCIR filter maximum gain will then be 12.2 dB as specified in CCIR Recommendation 468-4. This results in a 1 kHz unity gain frequency. If the RMS detector is selected, the unity gain frequency moves to 2 kHz as normally used in CCIR/ARM measurements. This is equivalent to the CCIR-2K selection on the Analog Analyzer. The display of the function meter does not depend upon the reading of the Level monitor.

See Also `AP.S1DSP.GenAnlr.FuncInput`, `AP.S1DSP.GenAnlr.FuncReady`, `AP.S1DSP.GenAnlr.FuncSettling`, `AP.S1DSP.GenAnlr.FuncTrig`, `AP.S1DSP.GenAnlr.RdgRate`

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.FuncRdg

Property

Syntax **`AP.S1DSP.GenAnlr.FuncRdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description
unit$	The following units (FFS, %FS, dBFS, Bits) are available for the following Function meter Modes: 2-Channel, THD+N Absolute, Bandpass, Noise-Unweighted, Noise-"A" Weighted, Noise-CCIR Weighted. The following units (%, dB, X/Y) are available for the following Function meter Modes: Ratio, Crosstalk, THD+N relative.

Description This command returns a settled reading for the Digital Domain Audio Tester Function meter and zeros the ready count.

See Also AP.S1DSP.GenAnlr.FuncInput,
AP.S1DSP.GenAnlr.FuncMode,
AP.S1DSP.GenAnlr.FuncReady,
AP.S1DSP.GenAnlr.FuncSettling,
AP.S1DSP.GenAnlr.FuncTrig

Example

```
Sub Main
   AP.File.OpenTest "GENANLR4.AT1"
   AP.S1DSP.GenAnlr.FuncSettling( 1, .00001, "FFS", _
      3, .1, 1)
   AP.S1DSP.GenAnlr.FuncTrig
   Do
      Ready1 = AP.S1DSP.GenAnlr.FuncReady
   Loop Until Ready1 > 0
   Reading1 = AP.S1DSP.GenAnlr.FuncRdg("FFS")
   NewLine$ = Chr(13)
   a$= "Function Meter "+Left(Str$(Reading1),6)+"FFS"
   AP.Prompt.Text = a$ + NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.GenAnlr.FuncReady

Property

Syntax AP.S1DSP.GenAnlr.FuncReady

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Domain Audio Tester 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 AP.S1DSP.GenAnlr.FuncRdg or AP.S1DSP.GenAnlr.FuncTrig commands will zero the ready count.

If the reading is found to be ready, a call to the AP.S1DSP.GenAnlr.FuncRdg command will be guaranteed to return quickly.

See Also AP.S1DSP.GenAnlr.FuncInput, AP.S1DSP.GenAnlr.FuncRdg, AP.S1DSP.GenAnlr.FuncSettling, AP.S1DSP.GenAnlr.FuncTrig

Example See example for AP.S1DSP.GenAnlr.FuncRdg.

AP.S1DSP.GenAnlr.FuncSettling

Method

Syntax **AP.S1DSP.GenAnlr.FuncSettling(***tolerance#, floor#, floorunit$, points%, delay#, algorithm%***)**

Description This command sets the settling parameters for the AP.S1DSP.GenAnlr.FuncRdg command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also AP.S1DSP.GenAnlr.FuncInput, AP.S1DSP.GenAnlr.FuncRdg, AP.S1DSP.GenAnlr.FuncReady, AP.S1DSP.GenAnlr.FuncTrig

Example See example for AP.S1DSP.GenAnlr.FuncRdg.

AP.S1DSP.GenAnlr.FuncTrig

Method

Syntax **AP.S1DSP.GenAnlr.FuncTrig**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.GenAnlr.FuncRdg` command. The reading in progress is aborted.

See Also `AP.S1DSP.GenAnlr.FuncInput`, `AP.S1DSP.GenAnlr.FuncRdg`, `AP.S1DSP.GenAnlr.FuncReady`, `AP.S1DSP.GenAnlr.FuncSettling`

Example See example for `AP.S1DSP.GenAnlr.FuncRdg`.

AP.S1DSP.GenAnlr.LevelRdg

Property

Syntax **`AP.S1DSP.GenAnlr.LevelRdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: dec, hex.

Description This command returns a settled reading for the Digital Domain Audio Tester Level meter and zeros the ready count.

See Also `AP.S1DSP.GenAnlr.LevelReady`, `AP.S1DSP.GenAnlr.LevelSettling`, `AP.S1DSP.GenAnlr.LevelTrig`

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.LevelReady

Property

Syntax **`AP.S1DSP.GenAnlr.LevelReady`**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Domain Audio Tester Level 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 `AP.S1DSP.GenAnlr.LevelRdg` or `AP.S1DSP.GenAnlr.LevelTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.GenAnlr.LevelRdg` 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 `AP.S1DSP.GenAnlr.LevelTrig` command.

See Also `AP.S1DSP.GenAnlr.LevelRdg`, `AP.S1DSP.GenAnlr.LevelSettling`, `AP.S1DSP.GenAnlr.LevelTrig`

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.LevelSettling ❶ Method

Syntax **`AP.S1DSP.GenAnlr.LevelSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.S1DSP.GenAnlr.LevelRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.S1DSP.GenAnlr.LevelRdg`, `AP.S1DSP.GenAnlr.LevelReady`, `AP.S1DSP.GenAnlr.LevelTrig`

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.LevelTrig

Method

Syntax `AP.S1DSP.GenAnlr.LevelTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.GenAnlr.LevelRdg` command. The reading in progress is aborted.

See Also `AP.S1DSP.GenAnlr.LevelRdg`, `AP.S1DSP.GenAnlr.LevelReady`, `AP.S1DSP.GenAnlr.LevelSettling`

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.Output

Property

Syntax `AP.S1DSP.GenAnlr.Output`

Data Type Boolean

True	On
False	Off

Description This command sets the Digital Domain Audio Tester channel A and B outputs to ON or OFF if they have been individually enabled by the `AP.S1DSP.GenAnlr.ChAOutput` and `AP.S1DSP.GenAnlr.ChBOutput` commands.

See Also `AP.S1DSP.GenAnlr.ChAOutput`, `AP.S1DSP.GenAnlr.ChBOutput`

Example See example for `AP.S1DSP.GenAnlr.Ampl`.

AP.S1DSP.GenAnlr.RdgRate

Property

Syntax `AP.S1DSP.GenAnlr.RdgRate`

Data Type Integer

0	Auto RMS: 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.
1	4/sec RMS
2	8/sec RMS
3	16/sec RMS
4	32/sec RMS
5	64/sec RMS
6	4/sec QPK

Description This command sets the Digital Domain Audio Tester detector response and controls the update rate (integration period) of all three meters of this program. Selections include Auto with RMS detection, fixed rates from 4/sec through 64/sec with the RMS detector, and a 4/sec rate with a quasi-peak detector which complies with CCIR recommendation 468-4 and earlier.

Example See example for `AP.S1DSP.GenAnlr.FilterHP`.

AP.S1DSP.GenAnlr.Wfm

Property

Syntax `AP.S1DSP.GenAnlr.Wfm`

Data Type Integer

0	Sine
1	EQ Sine

Description This command selects the Digital Domain Audio Tester waveform.

Example See example for `AP.S1DSP.GenAnlr.EqAmpl`.

User Notes

User Notes

User Notes

Narrow Bandpass Filter

AP.S1DSP.Harmonic.AmplRdg

Property

Syntax **AP.S1DSP.Harmonic.AmplRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	The following units are available (V, dBV, dBu, dBrm, dBm) for the AP.S1DSP.Harmonic.Source command Anlr Reading Ampl mode and (%, X/Y, and dB) for the Anlr Reading Ratio mode.

Description This command returns a settled reading for the Narrow Bandpass Filter filtered amplitude meter and zeros the ready count.

See Also AP.S1DSP.Harmonic.AmplReady, AP.S1DSP.Harmonic.AmplSettling, AP.S1DSP.Harmonic.AmplTrig, AP.S1DSP.Harmonic.Source

Example

```
Sub Main
   AP.File.OpenTest "HARMONC2.AT1"  'Opens test
   With AP.S1Dsp.Harmonic
      .RdgRate = 0   'Set Reading Rate to Auto
      .AmplSettling(1, .0001, "%", 3, .03, 1)
      .FreqSettling(.5, .0002, "Hz", 3, .03, 1)
      .AmplTrig   'Trigger new Amplitude reading
      Do
      Ready1 = .AmplReady
      Loop Until Ready1 > 0   'Wait for new reading
      Reading1 = .AmplRdg("dB")  'Get new reading
      .FreqTrig   'Trigger new Frequency reading
      Do
      Ready2 = .FreqReady
      Loop Until Ready2 > 0   'Wait for new reading
      Reading2 = .FreqRdg("Hz")  'Get new reading
   End With
   NewLine$ = Chr(13)
```

```
   a$= "Filtered Amplitude " & Left(Str$(Reading1),6) _
      & "dB"
   b$= "Filter Frequency " & Left(Str$(Reading2),6) _
      & "Hz"
   AP.Prompt.Text = a$ & NewLine$ & b$ + NewLine
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S1DSP.Harmonic.AmplReady

Property

Syntax `AP.S1DSP.Harmonic.AmplReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Narrow Bandpass Filter filtered amplitude 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 `AP.S1DSP.Harmonic.AmplRdg` or `AP.S1DSP.Harmonic.AmplTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.Harmonic.AmplRdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.Harmonic.AmplRdg`, `AP.S1DSP.Harmonic.AmplSettling`, `AP.S1DSP.Harmonic.AmplTrig`

Example See example for `AP.S1DSP.Harmonic.AmplRdg`.

AP.S1DSP.Harmonic.AmplSettling ❶ Method

Syntax **AP.S1DSP.Harmonic.AmplSettling(***tolerance#, floor#, floorunit$, points%, delay#, algorithm%***)**

Description This command sets the settling parameters for the AP.S1DSP.Harmonic.AmplRdg command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also AP.S1DSP.Harmonic.AmplRdg, AP.S1DSP.Harmonic.AmplReady, AP.S1DSP.Harmonic.AmplTrig

Example See example for AP.S1DSP.Harmonic.AmplRdg.

AP.S1DSP.Harmonic.AmplTrig ❶ Method

Syntax **AP.S1DSP.Harmonic.AmplTrig**

Description Causes a restart of the reading cycle and zeros the ready count for the AP.S1DSP.Harmonic.AmplRdg comand. The reading in progress is aborted.

See Also AP.S1DSP.Harmonic.AmplRdg, AP.S1DSP.Harmonic.AmplReady, AP.S1DSP.Harmonic.AmplSettling

Example See example for AP.S1DSP.Harmonic.AmplRdg.

AP.S1DSP.Harmonic.FilterFreq ❶ Property

Syntax **AP.S1DSP.Harmonic.FilterFreq(***unit$***)**

Data Type Double

Parameters

Name	Description

unit$	String that designates the desired unit. The following units are valid for this command: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, PPM.

Description This command sets the Narrow Bandpass Filter filter frequency when the Filter Tuning Source `AP.S1DSP.Harmonic.FilterTuningSource` is set to Panel Setting.

It is important to remember that the tunable filter center frequency is limited to the range of 20 Hz to 20 kHz at the 48 kHz sample rate and to the range of 80 Hz to 80 kHz at the 192 kHz sample rate. A combination of Panel entry values and Harmonic or Freq Offset values can lead to commands to the filter which fall outside those ranges and error messages will result.

See Also AP.

Example See example for `AP.S1DSP.Harmonic.Source`.

AP.S1DSP.Harmonic.FilterTuning

Property

Syntax **`AP.S1DSP.Harmonic.FilterTuning`**

Data Type Integer

0	Pannel Setting: the value entered in the Filter Freq field is the tuning source. This is a fixed value when no sweep is taking place. If Harmonic is the selected Instrument and Filter Freq is the selected parameter at Source 1 or Source 2 setting parameter on the Sweep panel, then the Panel value and thus the filter center frequency can be swept as part of test.
1	Analyzer Filter: the present center frequency of the Analog Analyzer bandpass/bandreject filter is the tuning source. The Analog Analyzer BP/BR filter may in turn be tracking the Analog Generator frequency (Gen Track), be controlled by the Analog Analyzer Frequency Counter (Auto), or be Fixed at a value entered on the Analog Analyzer panel. If the Analog Analyzer BP/BR filter is in Auto mode, this selection permits selective measurements with HARMONIC which track the

	frequency of a signal from a test disc or test tape or which originate at some remotely located sweeping generator.
2	Generator Frequency: the present frequency of the main sinewave oscillator in the Analog Generator is the tuning source.

Description This command sets the Narrow Bandpass Filter Filter Tuning Source.

Example See example for `AP.S1DSP.Harmonic.Source`.

AP.S1DSP.Harmonic.FilterTuningMode ❶ Property

Syntax **`AP.S1DSP.Harmonic.FilterTuningMode`**

Data Type Integer

0	Direct: the tunable filter center frequency will be exactly equal to the frequency value of the selected Filter Tuning Source.
1	Harmonic: the tunable filter center frequency will be an integer multiple of the frequency value of the selected Tuning Source. The value of the integer is selectable from 2 through 9 in the Harmonic field.
2	Offset: the tunable filter center frequency will be higher than the frequency value of the selected Tuning Source by the value in the Freq Offset field if that value is positive, and will be lower than the source by the Freq Offset value if that value is negative.

Description This command sets one of three relationships between the bandpass filter center frequency `AP.S1DSP.Harmonic.FilterFreq` and the frequency value of that Tuning Source `AP.S1DSP.Harmonic.TuningSource`.

Example See example for `AP.S1DSP.Harmonic.Source`.

AP.S1DSP.Harmonic.FilterType ❶ Property

Syntax **`AP.S1DSP.Harmonic.FilterType`**

Data Type Integer

0	Highpass: the Digital Analyzer has flat frequency response across the range from 20 Hz to 21.75 kHz at the 48 kHz sample rate and from 80 Hz to 80 kHz at the 192 kHz sample rate.
1	BP-Wide: the Q of 12 (1/8 octave) filter is selected. This filter is available both at 48 kHz and 192 kHz sample rates (80 kHz maximum center frequency).
2	BP-Narrow: the Q of 15 (1/10 octave) filter is selected. This filter is available only at the 48 kHz sample rate (20 kHz maximum center frequency).

Description This command sets the Narrow Bandpass Filter filter type.

Example See example for `AP.S1DSP.Harmonic.Source`.

AP.S1DSP.Harmonic.FreqOffset

Property

Syntax **`AP.S1DSP.Harmonic.FreqOffset(`*`unit$`*`)`**

Data Type Double

It is important to remember that the tunable filter center frequency is limited to the range of 20 Hz to 20 kHz at the 48 kHz sample rate and the 80 Hz to 80 kHz range at the 192 kHz sample rate. High positive Freq Offset values combined with high values of the Tuning Source presently selected, or high negative Freq Offset values combined with low values of the Tuning Source can lead to filter center frequencies which cannot be achieved. For example, if the Filter Tuning Source is Analog Generator and if the Analog Generator frequency is set or swept below 1 kHz, a Freq Offset value of -1 kHz will cause an error message.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, PPM.

Description This command sets the Narrow Bandpass Filter offset frequency.

The Freq Offset field permits user entry of a number which controls the frequency difference between the bandpass filter center frequency and the selected Filter Tuning Source when the Filter Tuning Mode

`AP.S1DSP.Harmonic.FilterTuningMode` is set to Offset. It has no effect if the Filter Tuning Mode is not set to Offset. A positive value in this field causes the filter center frequency to be higher than the source by the specified value. A negative value causes the filter frequency to be lower than the source by the specified value.

See Also

Example See example for `AP.S1DSP.Harmonic.Source`.

AP.S1DSP.Harmonic.FreqRdg

Property

Syntax **`AP.S1DSP.Harmonic.FreqRdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description
`unit$`	String that designates the desired unit. The following units are valid for this command: Hz, F/R, %Hz, cent, octs, decs, d%, PPM.

Description This command returns a settled reading for the Narrow Bandpass Filter filter frequency meter and zeros the ready count.

See Also `AP.S1DSP.Harmonic.FreqReady`, `AP.S1DSP.Harmonic.FreqSettling`, `AP.S1DSP.Harmonic.FreqTrig`

Example See example for `AP.S1DSP.Harmonic.AmplRdg`.

AP.S1DSP.Harmonic.FreqReady

Property

Syntax **`AP.S1DSP.Harmonic.FreqReady`**

Data Type Integer

`0`	Reading not ready.
`>0`	Reading ready.

Description This command returns the Narrow Bandpass Filter filter 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 `AP.S1DSP.Harmonic.FreqRdg` or `AP.S1DSP.Harmonic.FreqTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.Harmonic.FreqRdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.Harmonic.FreqRdg`, `AP.S1DSP.Harmonic.FreqSettling`, `AP.S1DSP.Harmonic.FreqTrig`

Example See example for `AP.S1DSP.Harmonic.AmplRdg`.

AP.S1DSP.Harmonic.FreqSettling ❶ Method

Syntax **`AP.S1DSP.Harmonic.FreqSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.S1DSP.Harmonic.FreqRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.S1DSP.Harmonic.FreqRdg`, `AP.S1DSP.Harmonic.FreqReady`, `AP.S1DSP.Harmonic.FreqTrig`

Example See example for `AP.S1DSP.Harmonic.AmplRdg`.

AP.S1DSP.Harmonic.FreqTrig

Method

Syntax `AP.S1DSP.Harmonic.FreqTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.Harmonic.FreqRdg` comand. The reading in progress is aborted.

See Also `AP.S1DSP.Harmonic.FreqRdg`, `AP.S1DSP.Harmonic.FreqReady`, `AP.S1DSP.Harmonic.FreqSettling`

Example See example for `AP.S1DSP.Harmonic.AmplRdg`.

AP.S1DSP.Harmonic.RdgRate

Property

Syntax `AP.S1DSP.Harmonic.RdgRate`

Data Type Integer

0	Auto: this selection makes an automatic selection of the detector reading rate as a function of the present filter center frequency. During a sweep, the Auto selection chooses the fastest reading rate which does not compromise the specified accuracy of the detector. When a sweep involving the HARMONIC analyzer is not in progress, Auto sets the reading rate to 4/sec.
1	4/sec.
2	8/sec.
3	16/sec.
4	32/sec.
5	64/sec.

Description This command sets the Narrow Bandpass Filter detector integration period and RMS detector reading rate. When any of the specific reading rates are selected, the detector reading rate remains fixed at that value regardless of the filter center frequency. This can result in measurement errors when using fast reading rates at low filter frequencies.

Example See example for `AP.S1DSP.Harmonic.Source`.

AP.S1DSP.Harmonic.Source

Property

Syntax **`AP.S1DSP.Harmonic.Source`**

Data Type Integer

0	Anlr Reading Ampl: provides signals with an absolute amplitude calibration in units such as Volts, dBV, dBr, dBu, and dBm.
1	Anlr Reading Ratio: provides calibration of Reading meter signals referred to the amplitude of another signal. This includes individual harmonic amplitude measurements in % or dB below the fundamental when the Analog Analyzer Function meter is in THD+N Ratio function, or measurements of individual IMD product amplitudes in % or dB when the Reading meter is in one of the IMD functions.

Description This command sets the Narrow Bandpass Filter A/D input source.

Example

```
Sub Main
   AP.File.OpenTest "HARMONC1.AT1"  'Open test
   With AP.S1Dsp.Harmonic
      .Source = 0              'Anlr reading ratio
      .FilterTuningMode = 1    'Harmonic tuning mode
      .FilterTuning = 1        'Select analyzer filter
      .Value = 1               'Set Harmonic to 1
      .FreqOffset("Hz") = 0    'Set Hz offset
      .FilterFreq("Hz") = 1000   'Set filter freq 1k
      .FilterType = 1          'Set filter Type bp-wide
      .RdgRate = 0             'Set Reading Rate
      AP.Sweep.Start
      AP.Sweep.Source1.Start("Hz") = 4000
      .Value = 2               'Set Harmonic to 2
      AP.Sweep.Start
      AP.Sweep.Source1.Start("Hz") = 6000
      .Value = 3               'Set Harmonic to 3
   End With
   AP.Sweep.Start
```

```
End Sub
```

AP.S1DSP.Harmonic.Value

Property

Syntax `AP.S1DSP.Harmonic.Value`

Data Type Double

Description This command sets the Narrow Bandpass Filter harmonic value.

See Also `AP.S1DSP.Harmonic.FilterTuningMode`

Example See example for `AP.S1DSP.Harmonic.Source`.

User Notes

User Notes

User Notes

User Notes

User Notes

Quasi-Anechoic Acoustical Tester/Generator

AP.S1DSP.MLS.Ampl

Property

Syntax **AP.S1DSP.MLS.Ampl(*unit$*)**

Data Type Double — Valid amplitude settings are from 0 to 100 %FS.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dBFS, Bits

Description This command sets the Quasi-Anechoic Acoustical Tester Digital Generator Amplitude.

Example

```
Sub Main
   AP.File.OpenTest “MLS1.AT1"    ‘Opens test
   With AP.S1Dsp.Mls
      .InputFormat = 0            ‘Set Input to A/D
      .Ampl(“FFS”) = 1  ‘Set DGEN Amplitued to 1.000 FFS
      .ChAOutput = True           ‘Set Ch A output on
      .ChBOutput = True           ‘Set Ch B output on
      .Output = True              ‘Set DGEN on
      .Ch1Source = 3              ‘Set Anlr ratio
      .Ch2Source = 3              ‘Set Anlr ratio
      Wait 1.5
      .Ch1Trig                    ‘Trigger Ch 1 reading
      Do
         Ready1 = .Ch1Ready       ‘Check status
         Loop Until Ready1 > 0
      Reading1 = .Ch1Rdg(“FFS”)   ‘Get reading
      .Ch2Trig                    ‘Trigger Ch 2 reading
      Do
         Ready2 = .Ch2Ready       ‘Ckeck status
         Loop Until Ready2 > 0
      Reading2 = .Ch2Rdg(“FFS”)   ‘Get reading
   End With
   NewLine$ = Chr(13)
```

```
    a$= "Ch1 Peak Mon " & Left(Str$(Reading1),6) & "FFS"
    b$= "Ch2 Peak Mon " & Left(Str$(Reading2),6) & "FFS"
    AP.Prompt.Text = a$ & NewLine$ & b$ & NewLine
    AP.Prompt.ShowWithContinue
    Beep
    Stop
End Sub
```

AP.S1DSP.MLS.Ch1Rdg

Property

Syntax **AP.S1DSP.MLS.Ch1Rdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	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 Quasi-Anechoic Acoustical Tester channel 1 Peak Monitor meter and zeros the ready count.

See Also AP.S1DSP.MLS.Ch1Ready, AP.S1DSP.MLS.Ch1Trig

Example See example for AP.S1DSP.MLS.Ampl.

AP.S1DSP.MLS.Ch1Ready

Property

Syntax **AP.S1DSP.MLS.Ch1Ready**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Quasi-Anechoic Acoustical Tester channel 1 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 `AP.S1DSP.MLS.Ch1Rdg` or `AP.S1DSP.MLS.Ch1Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.MLS.Ch1Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.MLS.Ch1Rdg, AP.S1DSP.MLS.Ch1Trig`

Example See example for `AP.S1DSP.MLS.Ampl`.

AP.S1DSP.MLS.Ch1Source

Property

Syntax **`AP.S1DSP.MLS.Ch1Source`**

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.MLS.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Gen-Mon
5	DSP A
6	DSP B
7	None

The following list contains the selections relevant to the `AP.S1DSP.MLS.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Quasi-Anechoic Acoustical Tester Channnel 1 Input.

Example See example for `AP.S1DSP.MLS.Ampl`.

AP.S1DSP.MLS.Ch1Trig

Method

Syntax **`AP.S1DSP.MLS.Ch1Trig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.MLS.Ch1Rdg` command. The reading in progress is aborted.

See Also `AP.S1DSP.MLS.Ch1Rdg, AP.S1DSP.MLS.Ch1Ready`

Example See example for `AP.S1DSP.MLS.Ampl`.

AP.S1DSP.MLS.Ch2Rdg

Property

Syntax **`AP.S1DSP.MLS.Ch2Rdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description
`unit$`	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 Quasi-Anechoic Acoustical Tester channel 2 Peak Monitor meter and zeros the ready count.

See Also `AP.S1DSP.MLS.Ch2Ready, AP.S1DSP.MLS.Ch2Trig`

Example See example for `AP.S1DSP.MLS.Ampl`.

AP.S1DSP.MLS.Ch2Ready

Property

Syntax **`AP.S1DSP.MLS.Ch2Ready`**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Quasi-Anechoic Acoustical Tester channel 2 Peak Monitor meter 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 `AP.S1DSP.MLS.Ch2Rdg` or `AP.S1DSP.MLS.Ch2Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.MLS.Ch2Rdg` command will be guaranteed to return quickly.

See Also `AP.S1DSP.MLS.Ch2Rdg, AP.S1DSP.MLS.Ch2Trig`

Example See example for `AP.S1DSP.MLS.Ampl`.

AP.S1DSP.MLS.Ch2Source

Property

Syntax **`AP.S1DSP.MLS.Ch1Source`**

Data Type Integer

The following list contains the selections relevant to the `AP.S1DSP.MLS.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Gen-Mon
5	DSP A
6	DSP B
7	None

The following list contains the selections relevant to the `AP.S1DSP.MLS.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Quasi-Anechoic Acoustical Tester Channnel 2 Input.

Example See example for `AP.S1DSP.MLS.Ampl`.

AP.S1DSP.MLS.Ch2Trig

Method

Syntax `AP.S1DSP.MLS.Ch2Trig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.MLS.Ch2Rdg` comand. The reading in progress is aborted.

See Also `AP.S1DSP.MLS.Ch2Rdg, AP.S1DSP.MLS.Ch2Ready`

Example See example for `AP.S1DSP.MLS.Ampl`.

AP.S1DSP.MLS.ChAOutput

Property

Syntax `AP.S1DSP.MLS.ChAOutput`

Data Type Boolean

True	ON.
False	OFF.

Description This command sets Quasi-Anechoic Acoustical Tester Generator Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.MLS.ChBOutput`

Example See example for `AP.S1DSP.MLS.Ampl`.

AP.S1DSP.MLS.ChBOutput

Property

Syntax `AP.S1DSP.MLS.ChBOutput`

Data Type Boolean

True	ON.
False	OFF.

Description This command sets the Quasi-Anechoic Acoustical Tester Output A to ON or OFF.

The command returns a TRUE if the output is ON and FALSE if the output is OFF.

See Also `AP.S1DSP.MLS.ChAOutput`

Example See example for `AP.S1DSP.MLS.Ampl`.

AP.S1DSP.MLS.InputFormat

Property

Syntax `AP.S1DSP.MLS.InputFormat`

Data Type Integer

0	A/D
1	Digital

Description This command sets the Quasi-Anechoic Acoustical Tester Input Format.

Example See example for `AP.S1DSP.MLS.Ampl`.

AP.S1DSP.MLS.Output

Property

Syntax `AP.S1DSP.MLS.Output`

Data Type Boolean

True	ON

False	OFF

Description This command sets the Quasi-Anechoic Acoustical Tester channel A and B outputs to ON or OFF if they have been individually enabled by the `AP.S1DSP.MLS.ChAOutput` and `AP.S1DSP.MLS.ChBOutput` commands.

See Also `AP.S1DSP.MLS.ChAOutput, AP.S1DSP.MLS.ChBOutput`

Example See example for `AP.S1DSP.MLS.Ampl`.

AP.S1DSP.MLS.Sequence

Property

Syntax **`AP.S1DSP.MLS.Sequence`**

Data Type Integer

0	Pink Noise #1
1	Pink Noise #2
2	Pink Noise #3
3	Pink Noise #4
4	White Noise #1

Description This command sets the Quasi-Anechoic Acoustical Tester Input MLS Sequence.

The MLS Sequence field selects from among four different MLS sequences to avoid interference when several acoustic test stations are operating near one another. The four pink-noise-filtered 32k point maximum length sequences are numbered Pink Noise #1 through Pink Noise #4. Each will cross-correlate to approximately -45 dB against any of the other three. These four sequences are all weighted with a pink noise filter to increase their low frequency energy and provide a constant power per octave across the audio band. This greatly improves the signal-to-noise ratio at low frequencies, increasing measurement accuracy in typical room ambient noise conditions. A single white noise sequence is also provided, labeled White Noise #1, for unusual applications where the large high frequency energy level may be desired and signal-to-noise ratio is not a concern. If the sequence is recorded on RDAT or other digital tape for later

measurement, it is important that the same sequence number be selected on playback. Otherwise no impulse response will be obtained.

Example

```
Sub Main
   AP.File.OpenTest "mls2.at1"   'Open test
   With AP.S1Dsp.Mls
      .TimeDelay("sec") = 0   'Set Time Delay to 0 sec
      .WindowStart = 1  'Set Time Start Window to <5%
      .WindowStop = 3   'Set Time Stop Window to <20%
      .TimeDisplay = 0  'Set Time Domain Display to _
         Impulse-Response
      .WfmDisplay = 0   'Set Wave Display to Interpolate
      .WindowETime = 0  'Set Energy-Time Window to _
         No Window
      .Sequence = 1  'Set Mls Sequence to Pink Noise #2
   End With
   AP.Sweep.Start
End Sub
```

AP.S1DSP.MLS.TimeDelay

Property

Syntax **AP.S1DSP.MLS.TimeDelay(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: sec, feet, meters

Description This command sets the Quasi-Anechoic Acoustical Tester Time Delay.

The Time Delay field is used to tell the DSP the distance from the speaker under test to the measurement microphone as a reference for the phase measurements. This information allows the DSP to subtract out the transit time delay from the phase readings. As the Time Delay value is adjusted the phase response will slope up or down reflecting the constant time delay component of the data. The initial value of Time Delay may be estimated from a measurement of the distance between loudspeaker and microphone. The proper final Time Delay

value may be determined experimentally as the peak amplitude on a time domain graph or to obtain the smallest slope on phase.

See Also AP.

Example See example for AP.S1DSP.MLS.Sequence.

AP.S1DSP.MLS.TimeDisplay

Property

Syntax `AP.S1DSP.MLS.TimeDisplay`

Data Type Integer

0	Impulse Response: will show the results of the MLS correlation which is the actual impulse response of the device under test.
1	Energy-Time: will display what is commonly called an energy-time curve. The energy-time curve computation process involves transforming the impulse response to the frequency domain, doing further processing in the frequency domain, and transforming the result back to the time domain. A frequency window may be used for the conversion from frequency domain back to time domain. The frequency window is selected in the Energy-Time Window field.

Description This command sets the Quasi-Anechoic Acoustical Tester Time Domain Display type.

Example See example for AP.S1DSP.MLS.Sequence.

AP.S1DSP.MLS.WfmDisplay

Property

Syntax `AP.S1DSP.MLS.WfmDisplay`

Data Type Integer

0	Interpolate
1	Display Samples
2	Peak Values

Description This command sets the Quasi-Anechoic Acoustical Tester waveform display mode.

When Interpolate is selected, the DSP will compute the data value, interpolated from the nearby measured values. This smooths out the stair-step appearance of frequency response curves at low frequencies with a Log horizontal axis, where the bin width (usually 2.93 Hz at the 48 kHz sample rate) occupies a significant portion of the screen.

When Display Samples is selected, the DSP will return the closest actual measured value without altering the data. Normal is the recommended display mode for frequency response data with a Linear horizontal axis or with a Log axis above 100 to 300 Hz. In these cases, the jagged lines caused by the FFT bin width are not usually noticeable.

When Peak Values is selected, The Peak mode will return the largest value between the last requested sweep point and the current one. Peak is recommended for time domain MLS displays (Impulse Response and Energy-Time). Peak mode would not normally be used for frequency response displays with MLS.AZ1, since high values are of no more interest than low values when plotting frequency response.

Example See example for `AP.S1DSP.MLS.Sequence`.

AP.S1DSP.MLS.WindowETime

Property

Syntax `AP.S1DSP.MLS.WindowETime`

Data Type Integer

0	No Window: will perform the required transformations with all frequency components of the signal included in the computations.
1	Half Hann: reduces the contribution of high frequencies. The low frequency information remains unchanged. When operating at the 48 kHz sample rate this window filters out energy above 12 kHz.
2	Hann: reduces both high and low frequency energy, concentrating on arrivals at the center of the frequency range. Since the processing occurs on a linear frequency scale, this will focus analysis on signals around one quarter of the

sample rate. At 48 kHz this will result in the 12 kHz energy dominating the energy-time display. This selection is not fundamentally useful for most applications, but is included for correlation to measurements by other manufacturers equipment where this window is used.

3	<240Hz >8kHz: filters out energy below 240 Hz and above 8 kHz, producing equal sensitivity to signals over a 5 octave range.
4	<124Hz >16kHz: spreads the analysis over a 7 octave range.

Description This command sets the Quasi-Anechoic Acoustical Tester Energy-Time Window selection.

Example See example for `AP.S1DSP.MLS.Sequence`.

AP.S1DSP.MLS.WindowStart

Property

Syntax **`AP.S1DSP.MLS.WindowStart`**

`None`

Data Type Integer

0	None:
1	<5%
2	<10%
3	<20%
4	<30%

Description This command sets the Quasi-Anechoic Acoustical Tester Start Time Window selection.

When a section of the impulse response (direct arrival signal before reflections, for example) is isolated and transformed into the frequency domain, the impulse amplitude at the beginning and ending of that section will generally not be exactly the same and thus will not splice smoothly. The sharp edges introduced into the impulse response by splicing unequal amplitudes will produce ripples in the resulting frequency response plot. Windowing the time domain data by attenuating the amplitude at the beginning and end of the section to be transformed will reduce this rippling, but also reduces the steepness of

transitions in the frequency response plots. The Time Start Window and Time Stop Window fields select the window applied to the impulse response (time domain) when transforming it to the frequency domain.

The time window is made up of two half-windows. The first half is selected in the Time Start Window field and is used to process the first portion of data, beginning at the Source 1 Start time on the Sweep panel. The second half-window is selected in the Time Stop Window field and processes the later portion of data, ending at the selected Stop time on the Sweep panel. Separate selection of the Source 1 Start and Stop half-windows permits creation of asymmetrical windows, which provide the optimum match to the asymmetrical shape of the typical impulse response. To change selections, click on the down arrow at the right of the field and click on the desired selection in the list which is displayed. The available selections at both the Time Start Window and Time Stop Window fields are a family of half-cycle raised cosine functions labeled NONE, <5%, <10%, <20% and <30%. The numeric value refers to the amount of the data record (time span multiplied by sample period) taken up by the window's transition from zero to full amplitude. The Time Start Window half-window starts with an amplitude of zero at the Sweep panel Start time and climbs to an amplitude of 1.00 (no attenuation) at or before the selected percentage of the record. The Time Stop Window half-window starts with an amplitude of 1.00 at or following a point during the record which is within the selected percentage of the record end, and falls to zero at the Sweep panel Stop time. The windows with a steeper transition will alter the data less but will also have less impact on the frequency response ripples. The more gradual transitions have greater ripple reduction but alter the data more.

Example See example for `AP.S1DSP.MLS.Sequence`.

AP.S1DSP.MLS.WindowStop

Property

Syntax `AP.S1DSP.MLS.WindowStop`

Data Type Integer

0 None:

1	<5%
2	<10%
3	<20%
4	<30%

Description This command sets the Quasi-Anechoic Acoustical Tester Stop Time Window selection.

See Also `AP.S1DSP.MLS.WindowStart`

Example See example for `AP.S1DSP.MLS.Sequence`.

User Notes

User Notes

User Notes

System One DSP Program

AP.S1DSP.Program

Property

Syntax `AP.S1DSP.Program`

Data Type Integer

0	None
1	Digital Domain Audio Tester (GENANLR): The Digital Domain Audio Tester (GENANLR) is a DSP-implemented digital-domain emulation of the most common analog domain types of audio measurement. Thus, it permits measurements on audio signals in the digital domain which can be readily compared to conventional analog audio measurements of other devices. GENANLR is not intended for measurements of analog-domain signals. GENANLR is a real-time program. Signal is continuously generated and measurements can be continuously observed in panel or bargraph mode for adjustment purposes. The signal generation capability of this program provides flexible sinewave generation at any amplitude and frequency. Digital domain signal measurement capabilities include frequency measurements, simultaneous level measurements on both channels, A-weighted or CCIR-468-weighted noise measurements with RMS or quasi-peak detector, and measurements through a bandreject (notch) or bandpass filter to support THD+N and selective measurements. Since these same types of signal generation and measurement are part of the capabilities of the analog hardware sections of System One, all these measurements may be made in any combinations of domain including cross-domain measurements to test A/D and D/A converters.
2	Spectrum Analyzer/Generator (FFTGEN): The DSP Generator and FFT Analyzer program (FFTGEN) consists of a real-time digital domain sinewave generator which alternately can generate signals from a downloaded

waveform file, and a general purpose signal acquisition, waveform display, and FFT spectrum analysis capability.

3 Spectrum Analyzer (FFTSLIDE): The Spectrum Analyzer program (FFTSLIDE) consists of a general purpose signal acquisition, waveform display, and FFT spectrum analysis capability featuring pre-trigger, the ability to slide the start point of an FFT transform anywhere in the record, flexible hardware triggering of acquisition, and the longest record length of any System One FFT program.

4 Multitone Analyzer/Generator (FASTTEST): The Multitone Generator/Analyzer program (FASTTEST) consists of a digital domain generator which can generate signals (usually multitone) from a downloaded waveform file, a signal acquisition, waveform display, and FFT spectrum analysis capability, and several forms of post-FFT processing to provide very rapid measurement of the frequency response, total distortion and noise, noise in the presence of test signal, and phase characteristics of an analog or digital audio device.

5 Triggered Multitone Tester (FASTTRIG): The Triggered Multitone Generator/Analyzer program (FASTTRIG) consists of:

A digital domain generator which can generate signals (usually multitone) from a downloaded waveform file.

A signal recognition and triggering feature which acquires signal into the analyzer only when it is a sufficiently close match to the waveform presently in the generator buffer.

A frequency error correction feature which uses the generator buffer waveform as a reference to correct for any frequency errors introduced when the multitone signal is recorded and reproduced or is furnished by a different generator operating from a crystal clock at a slightly different frequency.

A signal acquisition, waveform display, and FFT spectrum analysis capability, and several forms of post-FFT processing to provide very rapid measurement of the frequency response, total distortion and noise, noise in the presence of

test signal, and phase characteristics of an analog or digital audio device.

6 Quasi-Anechoic Acoustical Tester (MLS): The Quasi-Anechoic Acoustical Tester (MLS) program for the Digital Analyzer uses Maximum Length Sequence (MLS) testing to characterize the linear response of acoustical and electronic devices. It permits time-selective measurements in which one signal, such as the direct sound from a loudspeaker, may be separated from another similar signal, such as a room reflection. The time window may be adjusted to allow measurement of any arrival in a complex reverberation pattern. These signals may be examined in the time domain (showing energy as a function of time) or in the frequency domain (amplitude and phase vs frequency). Impulse responses may be saved to disk for later down-load to the DSP and further analysis.

Except in repetitive testing with unchanged dimensions between loudspeaker under test, measurement microphone, and reflecting surfaces, use of MLS typically involves both time domain and frequency domain displays. It is normally necessary to examine the time domain impulse response from MLS to determine the exact arrival time of the signal and the first reflection, designation of that time section for FFT spectrum analysis, and finally graphing of the anechoic frequency (and possibly phase) response for examination or comparison to limits.

7 Narrow Bandpass Filter (HARMONIC): The Harmonic Analyzer program is designed for analog input only, measuring signals from the Analog Analyzer Reading meter. The program consists of a sharp bandpass filter whose center frequency may be set or swept across the audio spectrum and above. Two bandwidths are available at center frequencies from 20 Hz to 20 kHz (48 kHz sample rate), the wider of which is also tunable from 80 Hz to 80 kHz at the 192 kHz sample rate. The wider filter, available at both sample rates, has a Q (ratio of center frequency to 3 dB bandwidth) of about 12 (1/8 octave bandwidth). The sharper filter, available only at the 48 kHz sample rate, has a Q of about 15 (1/10 octave bandwidth).

8 Digital Data Analyzer (BITTEST): This program generates imbedded digital audio test signals and measures the returned imbedded digital audio signals for bit errors. It can also control transmission of the Data Valid/Invalid bit. BITTEST operates at 48 kHz, 44.1 kHz, and 32 kHz sample rates. Only these three sample rates may be used with BITTEST. The imbedded audio test signal may be a pseudo-random sequence, constant valued samples ("digital dc"), a sinewave of selectable amplitude and frequency, or walking bit patterns. Generated word width in BITTEST is always 24 bits. Measurement word width is selectable for all signals, in the Resolution field of the Digital I/O panel. The measurement may display both real-time received data and errors in the received data sequence. Any amount of delay between transmitted and received signals is permissible, allowing testing of devices and transmission links with large amounts of delay or even recorder-reproducers. No dither is added to any of the signals.

The extensive signal generation and error measurement capability of BITTEST is useful for investigating the integrity of digital audio data links, recorders, etc. It is also invaluable for design test of digital interfaces. Each waveform in the program has a specific testing application. BITTEST operates only with digital domain input and output.

9 Codec Tester (CODEC): The FastTrig Coder Analyzer (CODEC) is designed for testing low bit rate coders and decoders which employ perceptual coding concepts to mask the higher levels of quantization noise and distortion caused by encoding the signal with much less information bandwidth than required for linear digital audio recording or transmission. CODEC is based on the FASTTRIG program with the addition of internal masking curve generation in the DSP program. CODEC is intended to be used with multitone signals which simulate complex program material. Following acquisition of the output of the companion decoder, CODEC generates a composite masking curve for the multitone signal received (based on some of the best current psychoacoustical research) which is used as a reference limit of audibility for artifacts. CODEC then integrates quantization noise and

distortion across critical bandwidths and compares those integrated measurements with the masking curve. Noise and distortion below the masking curve should be inaudible. Noise and distortion above the masking curve may be audible, especially to critical listeners and in listening environments which do not further mask the signal with ambient noise.

The FastTrig Coder Analyzer (CODEC) consists of:

A digital domain generator which can generate signals (usually multitone) from a downloaded waveform file.

A signal recognition and triggering feature which acquires signal into the analyzer only when it is a sufficiently close match to the waveform presently in the generator buffer.

A frequency error correction feature which uses the generator buffer waveform as a reference to correct for any frequency errors introduced when the multitone signal is recorded and reproduced or is furnished by a different generator operating from a crystal clock at a slightly different frequency.

A signal acquisition, waveform display, and FFT spectrum analysis capability, and several forms of post-FFT processing to provide masking curves for multitone signals, total distortion and noise, rapid measurement of the frequency response, noise in the presence of test signal, and phase characteristics of an analog or digital audio device.

Description This command selects a System One Digital Analyzer type.

Example

```
Sub Main
   System = AP.Application.SysType
   If System = 2 Then Debug.Print"This test is for _
      System One only."
   AP.App.NewTest
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 1
   AP.S1DSP.Program = 2       'Select FFTGen program
```

```
    AP.S1DSP.FFTGen.InputFormat = 0  'Select A/D input
    AP.S1DSP.FFTGen.FFTLength = 0 'Set FFT length to _
       Maximum
    AP.S1DSP.FFTGen.Window = 0 'Set FFT window to BH4
    AP.Sweep.Data1.Id = 6023   'Set sweep panel Data 1 _
       to Fftgen.Ch.1 Ampl
    AP.Sweep.Source1.Id = 5515 'Set sweep panel _
       Source 1 to Fftgen.FFT Freq
    AP.Sweep.Start
    AP.Sweep.Append = 1
    AP.S1DSP.FFTGen.Window = 1 'Set FFT window to Hann
    AP.Sweep.Retransform
    AP.S1DSP.FFTGen.Window = 2 'Set FFT window to Flat-Top
    AP.Sweep.Retransform
    AP.S1DSP.FFTGen.Window = 3 'Set FFT window to None
    AP.Sweep.Retransform
    AP.Data.OptimizeDisplay 0
End Sub
```

User Notes

User Notes

System Two Digital Input/Output

AP.S2Dio.ChAPeakRdg ❷ Property

Syntax **AP.S2Dio.ChAPeakRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	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 channel A Peak Monitor meter and zeros the ready count.

See Also AP.S2Dio.ChAPeakReady, AP.S2Dio.ChAPeakTrig

Example

```
'#Uses "Dioconst.apb"

Sub Main
   Dim rdgA As Double, rdgB As Double

   'S2Dio Peak Meter Sample code
   AP.S2Dio.InFormat = XLR_BAL    'XLR balanced input
   AP.S2Dio.InImpedance = Z110    'High impedance input
   AP.S2Dio.InMonitorMode = ABS_PEAK    'Measure _
      absolute peak
   AP.S2Dio.ChAPeakTrig           'Trigger channel A _
      peak meter
   AP.S2Dio.ChBPeakTrig           'Trigger channel A _
      peak meter
   Do Until (AP.S2Dio.ChAPeakReady And _
      AP.S2Dio.ChBPeakReady)
      'perform other actions while waiting for readings
      '...
   Loop
   rdgA = AP.S2Dio.ChAPeakRdg(FFS) 'Get channel A peak _
      reading
   rdgB = AP.S2Dio.ChBPeakRdg(FFS) 'Get channel A peak _
      reading
```

```
   AP.Prompt.Text = "Ch A = " & rdgA & " FFS" & _
      Chr(13) & "Ch B = " & rdgB & " FFS"
   AP.Prompt.ShowWithContinue
   Stop
End Sub
```

AP.S2Dio.ChAPeakReady ❷ Property

Syntax `AP.S2Dio.ChAPeakReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Input/Output channel A 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 `AP.S2Dio.ChAPeakRdg` or `AP.S2Dio.ChAPeakTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2Dio.ChAPeakRdg` command will be guaranteed to return quickly.

See Also `AP.S2Dio.ChAPeakRd, AP.S2Dio.ChAPeakTrig`

Example See example for `AP.S2Dio.ChAPeakRdg`.

AP.S2Dio.ChAPeakTrig ❷ Method

Syntax `AP.S2Dio.ChAPeakTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2Dio.ChAPeakRdg` command. The reading in progress is aborted.

See Also `AP.S2Dio.ChAPeakRdg, AP.S2Dio.ChAPeakReady`

Example See example for `AP.S2Dio.ChAPeakRdg`.

AP.S2Dio.ChBPeakRdg ❷ Property

Syntax **`AP.S2Dio.ChBPeakRdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description
`unit$`	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 channel B Peak Monitor meter and zeros the ready count.

See Also `AP.S2Dio.ChBPeakReady, AP.S2Dio.ChBPeakTrig`

Example See example for `AP.S2Dio.ChAPeakRdg`.

AP.S2Dio.ChBPeakReady ❷ Property

Syntax **`AP.S2Dio.ChBPeakReady`**

Data Type Integer

`0`	Reading not ready.
`>0`	Reading ready.

Description This command returns the Digital Input/Output channel B 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 `AP.S2Dio.ChBPeakRdg` or `AP.S2Dio.ChBPeakTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the AP.S2Dio.ChBPeakRdg command will be guaranteed to return quickly.

See Also AP.S2Dio.ChBPeakRdg, AP.S2Dio.ChBPeakTrig

Example See example for AP.S2Dio.ChAPeakRdg.

AP.S2Dio.ChBPeakTrig ❷ Method

Syntax **AP.S2Dio.ChBPeakTrig**

Description Causes a restart of the reading cycle and zeros the ready count for the AP.S2Dio.ChBPeakRdg comand. The reading in progress is aborted.

See Also AP.S2Dio.ChBPeakRdg, AP.S2Dio.ChBPeakReady

Example See example for AP.S2Dio.ChAPeakRdg.

AP.S2Dio.DelayRdg ❷ Property

Syntax **AP.S2Dio.DelayRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, PPM.

Description This command returns a settled reading for the Digital Input/Output Delay from Out meter and zeros the ready count.

See Also AP.S2Dio.DelayReady, AP.S2Dio.DelaySettling, AP.S2Dio.DelayTrig

Example

```
'#Uses "Dioconst.apb"

Sub Main
```

```
   Dim rdgA As Double

   'S2Dio Delay From Out meter sample code
   AP.S2Dio.DelaySettling 1.0, 100e-6, SEC, 1, 0.0, NONE
   AP.S2Dio.DelayTrig      'Trigger channel A peak meter
   Do Until AP.S2Dio.DelayReady
      'perform other actions while waiting for reading
      '...
   Loop
   rdgA = AP.S2Dio.DelayRdg(SEC) 'Get channel A _
      peak reading
   AP.Prompt.Text = "Delay = " & rdgA
   AP.Prompt.ShowWithContinue
   Stop
End Sub
```


AP.S2Dio.DelayReady

❷ Property

Syntax `AP.S2Dio.DelayReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Input/Output Delay from Out 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 `AP.S2Dio.DelayRdg` or `AP.S2Dio.DelayTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2Dio.DelayRdg` command will be guaranteed to return quickly.

See Also `AP.S2Dio.DelayRd, AP.S2Dio.DelaySettling, AP.S2Dio.DelayTrig`

Example See example for `AP.S2Dio.DelayRdg`.

AP.S2Dio.DelaySettling ❷ Method

Syntax **AP.S2Dio.DelaySettling(***tolerance#, floor#, floorunit$, points%, delay#, algorithm%***)**

Description This command sets the settling parameters for the AP.S2Dio.DelayRdg command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also AP.S2Dio.DelayRdg, AP.S2Dio.DelayReady, AP.S2Dio.DelayTrig

Example See example for AP.S2Dio.DelayRdg.

AP.S2Dio.DelayTrig ❷ Method

Syntax **AP.S2DSP.DelayTrig**

Description Causes a restart of the reading cycle and zeros the ready count for the AP.S2Dio.DelayRdg command. The reading in progress is aborted.

See Also AP.S2Dio.DelayRdg, AP.S2Dio.DelayReady, AP.S2Dio.DelaySettling

Example See example for AP.S2Dio.DelayRdg.

AP.S2Dio.FlagChAInvalidRdg ❷ Property

Syntax **AP.S2Dio.FlagChAInvalidRdg**

Data Type Boolean

True	Error
False	Proper operation

Description This reading returns the state of the channel A Validity bit. The CH A 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 AP.S2Dio.OutSendInvalid, AP.S2Dio.FlagChBInvalidRdg, AP.S2Dio.FlagInvalidRdg.

Example

```
Sub Main
   AP.Application.NewTest
   AP.Application.PanelOpen apbPanelDigIOSmall
   AP.S2Dio.InFormat = 3
   If AP.S2Dio.FlagInvalidRdg  = True Then _
      Debug.Print "Invalid Error"
   AP.Application.PanelOpen apbPanelDigIOLarge
   AP.S2Dio.OutSendInvalid = True
   Wait 1
   If AP.S2Dio.FlagConfidenceRdg = True Then _
      Debug.Print "Confidence Error"
   If AP.S2Dio.FlagLockRdg = True Then _
      Debug.Print "Lock Error"
   If AP.S2Dio.FlagCodingRdg = True Then _
      Debug.Print "Coding Error"
   If AP.S2Dio.FlagParityRdg = True Then _
      Debug.Print "Parity Error"
   If AP.S2Dio.FlagChAInvalidRdg = True Then
      Debug.Print "ChA Invalid Error"
   If AP.S2Dio.FlagChBInvalidRdg = True Then _
      Debug.Print "ChB Invalid Error"
End Sub
```


AP.S2Dio.FlagChBInvalidRdg

❷ Property

Syntax AP.S2Dio.FlagChBInvalidRdg

Data Type Boolean

True	Error
False	Proper operation

Description This reading returns the state of the channel B Validity bit. The CH B 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 `AP.S2Dio.OutSendInvalid`, `AP.S2Dio.FlagChAInvalidRdg`, `AP.S2Dio.FlagInvalidRdg`.

Example See example for `AP.S2Dio.FlagChAInvalidRdg`.

AP.S2Dio.FlagCodingRdg

❷ Property

Syntax **`AP.S2Dio.FlagCodingRdg`**

Data Type Boolean

True	Error
False	Proper operation

Description This reading returns the state of the channel B Validity bit. The Coding reading indicates a deviation from proper biphase coding in the input serial stream (ignoring preambles). Proper biphase 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 biphase 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 `AP.S2Dio.FlagChAInvalidRdg`.

AP.S2Dio.FlagConfidenceRdg

❷ Property

Syntax **`AP.S2Dio.FlagConfidenceRdg`**

Data Type Boolean

True	Error
False	Proper operation

Description 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.

Example See example for `AP.S2Dio.FlagChAInvalidRdg`.

AP.S2Dio.FlagInvalidRdg ❷ Property

Syntax **`AP.S2Dio.FlagInvalidRdg`**

Data Type Boolean

`True`	Error
`False`	Proper operation

Description This reading uses OR logic to determine if either the channel A or channel B Validity bit is set as defined in the Professional and Consumer standards. One Validity bit is sent in each subframe.

See Also `AP.S2Dio.OutSendInvalid`, `AP.S2Dio.FlagChAInvalidRdg`, `AP.S2Dio.FlagChBInvalidRdg`.

Example See example for `AP.S2Dio.FlagChAInvalidRdg`.

AP.S2Dio.FlagLockRdg ❷ Property

Syntax **`AP.S2Dio.FlagLockRdg`**

Data Type Boolean

`True`	Error

False	Proper operation

Description The Lock reading indicates when the digital input phase-locked loop is unable to lock to the incoming signal.

Example See example for `AP.S2Dio.FlagChAInvalidRdg`.

AP.S2Dio.FlagParityRdg ❷ Property

Syntax `AP.S2Dio.FlagParityRdg`

Data Type Boolean

True	Error
False	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 `AP.S2Dio.FlagChAInvalidRdg`.

AP.S2Dio.InDeEmp ❷ Property

Syntax `AP.S2Dio.InDeEmp`

Data Type Integer

0	Off
1	50/15us 0dB
2	50/15us + 10dB
3	J17 0dB
4	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 (+10 dB for 50/15us, +20 dB for J17) to compensate for the matching headroom allowances of the System Two Digital Generator preemphasis `AP.S2Dio.OutPreEmp` capability.

See Also `AP.S2Dio.OutPreEmp`

Example See example for `AP.S2Dio.OutFormat`.

AP.S2Dio.InFormat ❷ Property

Syntax **`AP.S2Dio.InFormat`**

Data Type Integer

0	XLR (Bal): Front panel XLR digital input connector, balanced
1	BNC (unbal): Front panel BNC digital input connector, unbalanced
2	Optical: Front panel Toslink optical input connector
3	Gen Mon: Digital Generator XLR or BNC output connector
4	XLR w/Eq: Front panel XLR with equalization for 100 meter cable roll-off
5	BNC w/EQ: Front panel BNC with equalization for 100 meter cable roll-off
6	XLR common: Center tap of digital input transformer vs ground
7	Serial: Rear-panel general-purpose serial input connector
8	Parallel: Rear-panel parallel input connector

Description This command sets the Digital Input/Output Input Format.

Example See example for `AP.S2Dio.ChAPeakRdg`.

AP.S2Dio.InImpedance ❷ Property

Syntax **`AP.S2Dio.InImpedance`**

Data Type Integer

The following list contains the selections relevant to the AP.S2Dio.InFormat command XLR (Bal), XLR w/EQ, and XLR Common selections.

0	Hi Z
1	110 Ohms

The following list contains the selections relevant to the AP.S1Dio.InFormat command BNC (unbal), Gen Mon, and BNC w/EQ selections.

0	Hi Z
1	75 Ohms

Description This command sets the Digital Input/Output Input Impedance based on the selection for the AP.S2Dio.InFormat command.

Example See example for AP.S2Dio.ChAPeakRdg.

AP.S2Dio.InJitterBW ❷ Property

Syntax **AP.S2Dio.InJitterBW**

Data Type Integer

0	50Hz to 100kHz
1	120Hz to 100kHz
2	700Hz to 100kHz
3	1200Hz to 100kHZ

Description This command sets the Digital Input/Output Input bandwidth of the Interface Jitter AP.S2Dio.JitterRdg meter.

Example

```
'#Uses "Dioconst.apb"

Sub Main
   Dim rdg As Double

   'S2Dio Jitter meter sample code
   'Set up Jitter output
   AP.S2Dio.OutJitterType = SINUSOIDAL
```

```
    AP.S2Dio.OutJitterAmpl(UI) = 5.0
    AP.S2Dio.OutJitterFreq(HZ) = 2e3
    'Set up Jitter input
    AP.S2Dio.InJitterMode = IUI   'Unit interval mode
    AP.S2Dio.InJitterDetector = AVG 'Average detector
    AP.S2Dio.InJitterBW = HZ50 '50Hz - 100kHz bandwidth
    AP.S2Dio.JitterSettling 5.0, 1e-6, UI, 1, 0.0, NONE
    AP.S2Dio.JitterTrig           'Trigger channel A _
       peak meter
    Do Until AP.S2Dio.JitterReady
       'perform other actions while waiting for reading
       '...
    Loop
    rdg = AP.S2Dio.JitterRdg(UI)  'Get channel A _
       peak reading
    AP.Prompt.Text = "Jitter = " & rdg & " UI"
    AP.Prompt.ShowWithContinue
    Stop
End Sub
```


AP.S2Dio.InJitterDetector ❷ Property

Syntax **AP.S2Dio.InJitterDetector**

Data Type Integer

0	Pk
1	Avg

Description This command selects the Digital Input/Output Input detector type for the Interface Jitter AP.S2Dio.JitterRdg meter.

See Also AP.S2Dio.

Example See example for AP.S2Dio.InJitterBW.

AP.S2Dio.InJitterMode ❷ Property

Syntax **AP.S2Dio.InJitterMode**

Data Type Integer

0	UI: Unit interval
1	Sec: Time

Description This command selects the measurement mode for the Digital Input/Output Interface Jitter `AP.S2Dio.JitterRdg` meter.

See Also `AP.S2Dio.`

Example See example for `AP.S2Dio.InJitterBW`.

AP.S2Dio.InMonitorMode ❷ Property

Syntax **`AP.S2Dio.InMonitorMode`**

Data Type Integer

0	Pos. Peak: causes the Level Monitors to display the most positive value during each measurement interval, which is approximately 1/4 second.
1	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).
2	Abs. Peak: causes display of the absolute value of the largest positive-going or negative-going value during each measurement interval.
3	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 `AP.S2Dio.ChAPeakRdg`.

AP.S2Dio.InResolution ❷ Property

Syntax **`AP.S2Dio.InResolution`**

Data Type Integer 8 to 24 bits.

Parameters None

Description This command sets the Digital Input/Output Input Resolution.

See Also `AP.S2Dio.OutResolution`

Example See example for `AP.S2Dio.OutFormat`.

AP.S2Dio.InScaleFreq ❷ Property

Syntax **`AP.S2Dio.InScaleFreq`**

Data Type Integer

0	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.
1	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.
2	Status Bits A: is the value of sample frequency encoded into the received channel A status bits.

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.

Example See example for `AP.S2Dio.OutFormat`.

AP.S2Dio.JitterRdg ❷ Property

Syntax **`AP.S2Dio.JitterRdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description

unit$	String that designates the desired unit. The following units are valid for this command: UI

Description This command returns a settled reading for the Digital Input/Output Interface Jitter meter and zeros the ready count.

See Also `AP.S2Dio.JitterReady, AP.S2Dio.JitterSettling, AP.S2Dio.JitterTrig`

Example See example for `AP.S2Dio.InJitterBW`.

AP.S2Dio.JitterReady ❷ Property

Syntax **`AP.S2Dio.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 `AP.S2Dio.JitterRdg` or `AP.S2Dio.JitterTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2Dio.JitterRdg` command will be guaranteed to return quickly.

See Also `AP.S2Dio.JitterRdg, AP.S2Dio.JitterSettling, AP.S2Dio.JitterTrig`

Example See example for `AP.S2Dio.InJitterBW`.

AP.S2Dio.JitterSettling ❷ Method

Syntax **AP.S2Dio.JitterSettling(***tolerance#, floor#, floorunit$, points%, delay#, algorithm%***)**

Description This command sets the settling parameters for the AP.S2Dio.JitterRdg command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also AP.S2Dio.JitterRdg, AP.S2Dio.JitterReady, AP.S2Dio.JitterTrig

Example See example for AP.S2Dio.JitterBW.

AP.S2Dio.JitterTrig ❷ Method

Syntax **AP.S2DSP.JitterTrig**

Description Causes a restart of the reading cycle and zeros the ready count for the AP.S2Dio.JitterRdg command. The reading in progress is aborted.

See Also AP.S2Dio.JitterRdg, AP.S2Dio.JitterReady, AP.S2Dio.JitterSettling

Example See example for AP.S2Dio.InJitterBW.

AP.S2Dio.OutCableSim ❷ Property

Syntax **AP.S2Dio.OutCableSim**

Data Type Boolean

True	Enable cable simulation.
False	Disable cable simulation.

Description This command enables or disables cable simulation.

A fixed hardware filter may be switched into the path to the XLR or BNC output connectors to simulate the effect of a typical 100-meter-long cable, to test the ability of a digital device under test to function with impaired signals. This feature is not available at the optical, general purpose serial, or parallel outputs. This cable simulation filter is approximately the inverse of the input cable equalization filter selectable as XLR w/EQ or BNC w/EQ in the Digital Input Format field, so the two should approximately compensate for one another when a short external cable is connected from Digital Output to Digital Input. However, there will still be some attenuation of the interface signal introduced by the cable simulation hardware. To switch the cable simulator in and out of the circuit:

Example See example for `AP.S2Dio.OutFormat`.

AP.S2Dio.OutCM ❷ Property

Syntax `AP.S2Dio.OutCM`

Data Type Boolean

True	Enable common mode output.
False	Disable common mode output.

Description This command enables or disables the Digital Input/Output Common Mode output.

Example See example for `AP.S2Dio.CMAmpl`.

AP.S2Dio.OutCMAmpl ❷ Property

Syntax `AP.S2Dio.OutCMAmpl(unit$)`

Data Type Double Range of values: 0.0 to 20.4

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: Vpp

Description This command sets the Digital Input/Output Common Mode Amplitude value.

Example

```
'#Uses "Dioconst.apb"
Const INPUT_VOLTAGE As Integer = 6102
Const COMMON_MODE_AMPL As Integer = 5317
Const COMMON_MODE_FREQ As Integer = 5318

Sub Main
   Dim reading As Double

   AP.Application.NewTest     'Reset panels
   AP.S2Dio.OutCMFreq(HZ) = 20e3
   AP.S2Dio.OutCMAmpl(VPP) = 1.0
   AP.S2Dio.OutCM = True
   AP.S2Dio.InFormat = XLR_COMMON
   AP.S2Dio.VoltageSettling 3.0, 10e-3, VPP, 3, 0.0, FLA
   AP.Sweep.CreateGraph = True
   AP.Sweep.Data1.Id = INPUT_VOLTAGE
   AP.Sweep.Data1.Top(V) = 8.0
   AP.Sweep.Data1.Bottom(V) = 0.0
   AP.Sweep.Source1.Id = COMMON_MODE_AMPL
   AP.Sweep.Source1.Start(VPP) = 0.0
   AP.Sweep.Source1.Stop(VPP) = 20.0
   AP.Sweep.Source1.Steps = 10
   AP.Sweep.Start
End Sub
```


AP.S2Dio.OutCMFreq ❷ Property

Syntax **AP.S2Dio.OutCMFreq(*unit$*)**

Data Type Double Range of values: 20.0 Hz to 40.0 kHz

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: Vpp

Description This command sets the Digital Input/Output Common Mode signal (sinewave) Frequency value.

Example See example for `AP.S2Dio.CMAmpl`.

AP.S2Dio.OutDitherType ❷ Property

Obsolete Obsolete command not recomended for new design.

Syntax **`AP.S2Dio.OutDitherType`**

Data Type Integer

0	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.
1	Rectangular: probability function dither provides the best signal to noise due to its one-half LSB amplitude, but suffers from modulation noise effects.
2	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.
3	None:

Description This command sets the Digital Input/Output Dither Type.

Dither amplitude is automatically set corresponding to the LSB of the value selected in the Output Resolution field or by the `AP.S2Dio.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 `AP.S2Dio.OutResolution` command.

AP.S2Dio.OutFormat ❷ Property

Syntax **AP.S2Dio.OutFormat**

Data Type Integer

0	XLR (bal)
1	BNC (unbal)
2	Optical
3	Serial
4	Parallel

Description This command sets the Digital Input/Output Output source.

Example

```
'#Uses "Dioconst.apb"

Const FIXED As Boolean = False
Const VARIABLE As Boolean = True
Const INTERVU As Integer = 3
Const INTERVU_AMPL As Integer = 6053
Const INTERVU_TIME As Integer = 5612
Const CHA_RCV_PREAMBLE As Integer = 0
Const INTERPOLATE As Integer = 0

Sub Main
   Dim rftime As Double

   AP.Application.NewTest  'Reset panels
   AP.Application.NewData        'Clear existing data
   'S2Dio Output sample code
   AP.S2Dio.OutRiseFall = VARIABLE 'Variable Rise/Fall _
      Time
   AP.S2Dio.OutRiseFallTime(SEC) = 16e-9 'Set Rise/Fall _
      Time
   AP.S2Dio.OutCableSim = False     'Ensure Cable _
      simulation off
   AP.S2Dio.OutFormat = XLR_BAL  'Output signal to DUT
   AP.S2Dio.OutRate(HZ) = 48  'Set 48 kHz sample rate
   AP.S2Dio.OutNoise = True   'Let's inject some noise
   AP.S2Dio.OutNoiseAmpl(VPP) = 0.1    '@ 0.1 Vppd
```

```
    AP.S2Dio.OutResolution = 20      '20-bit resolution
    AP.S2Dio.OutPreEmp = NO_EMPH  'No Pre-Emph
    AP.S2Dio.InDeEmp = NO_EMPH    'No De-Emph
    AP.S2Dio.InFormat = XLR_BAL   'Input signal from DUT
    AP.S2Dio.InResolution = 20    'Same resolution as _
       output
    AP.S2Dio.InScaleFreq = 1      'Scale Frequency by _
       measured Rate
    AP.S2DSP.Program = INTERVU       'Load Intervu DSP
    AP.S2DSP.Intervu.AmplVsTime = INTERPOLATE 'put in _
       Interpolate mode
    AP.S2DSP.Intervu.Trig = CHA_RCV_PREAMBLE 'Trigger _
       on Ch A Receive Preamble
    AP.Sweep.Source1.Id = INTERVU_TIME  'Sweep time
    AP.Sweep.Source1.Start(SEC) = 0.0 'start at 0.0 sec
    AP.Sweep.Source1.Stop(SEC) = 4e-6   'end at 4 nSec
    AP.Sweep.Source1.Steps = 255     '255 points
    AP.Sweep.Data1.Id = INTERVU_AMPL 'Measure amplitude
    AP.Sweep.Data1.Top(V) = 3.0   'Max amplitude of +2 V
    AP.Sweep.Data1.Bottom(V) = -3.0 'Min amplitude of -2 V
    AP.Sweep.Append = False       'Don't append first _
       waveform
    AP.S2Dio.OutCableSim = True         'Acquire with _
       simulated long cable
    AP.Sweep.Start
    AP.S2Dio.OutCableSim = False  'Turn off long cable _
       simulation
    AP.Sweep.Append = True     'Append any additional _
       waveforms
    AP.Sweep.Start                'Get a waveform
    For rftime = 50e-9 To 350e-9 Step 100e-9 'What _
       affect does changing
       AP.S2Dio.OutRiseFallTime(SEC) = rftime 'Set _
          Rise/Fall Time
       AP.Sweep.Start             'Append next sweep
    Next rftime
End Sub
```

Comment Cable simulation looks just like RiseFallTime = 350 nSec

AP.S2Dio.OutJitterAmpl ❷ Property

Syntax **AP.S2Dio.OutJitterAmpl(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: UI, sec.

Description This command sets the Digital Input/Output Jitter Amplitude value.

See Also AP.S2Dio.OutJitterEqCurve

Example See example for AP.S2Dio.InJitterBW.

AP.S2Dio.OutJitterEqCurve ❷ Method

Syntax **AP.S2Dio.OutJitterEqCurve(*filename$, column%*)**

Data Type Boolean

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN Eq file (.adq).
column%	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 Jitter Generator. Values in the file will be used as multiply factors in czlculating the Digital Input/Output Jitter Amplitude value.

See Also AP.S2Dio.OutJitterAmpl

AP.S2Dio.OutJitterFreq ❷ Property

Syntax `AP.S2Dio.OutJitterFreq(unit$)`

Data Type Double

Parameters

Name	Description
unit$	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 `AP.S2Dio.InJitterBW`.

AP.S2Dio.OutJitterType ❷ Property

Syntax `AP.S2Dio.OutJitterType`

Data Type Integer

0	Off
1	Sinusoidal
2	Lowpass Random
3	Squarewave
4	Wideband Random

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 `AP.S2Dio.InJitterBW`.

AP.S2Dio.OutNoise ❷ Property

Syntax `AP.S2Dio.OutNoise`

Data Type Boolean

True	Enable noise output.
False	Disable noise output.

Description This command enables or disables the Digital Input/Output Interfering Noise output.

See Also `AP.S2Dio.OutNoiseAmpl`

AP.S2Dio.OutNoiseAmpl ❷ Property

Syntax **`AP.S2Dio.OutNoiseAmpl(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
`unit$`	String that designates the desired unit. The following units are valid for this command: Vpp

Description This command sets the Digital Input/Output Noise Amplitude value.

Example See example for `AP.S2Dio.OutFormat`.

AP.S2Dio.OutPreEmp ❷ Property

Syntax **`AP.S2Dio.OutPreEmp`**

Data Type Integer

0	Off
1	50/15us 0dB
2	50/15us - 10dB
3	J17 0dB
4	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 `AP.S2Dio.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 `AP.S2Dio.InDeEmp`

Example See example for `AP.S2Dio.OutFormat`.

AP.S2Dio.OutRate ❷ Property

Syntax **`AP.S2Dio.OutRate(`*`unit$`*`)`**

Data Type Double

The digital output sample rate of System Two may be freely set across the range from 29 kHz to 52 kHz.

Parameters

Name	Description
`unit$`	String that designates the desired unit. The following units are valid for this command: Hz

Description This command sets the Digital Output Sample Rate Frequency value.

The Rate field also appears labeled as Internal Sample Rate on the Analog Generator panel. Any of these rate fields may be used for entry with the new value becoming immediately visible on the other panels. Note that the rate set in these fields is also used as the clock rate for the Low Bandwidth A/D converters, and four times this rate is used for the High Bandwidth A/D converters.

Example See example for `AP.S2Dio.OutFormat`.

AP.S2Dio.OutResolution ❷ Property

Syntax `AP.S2Dio.OutResolution`

Data Type Integer — The width or resolution of the imbedded digital audio signal may be set to any value from 8 to 24 bits.

Parameters None

Description This command sets the Digital Output Resolution.

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.

See Also `AP.S2Dio.InResolution`

Example See example for `AP.S2Dio.OutFormat`.

AP.S2Dio.OutRiseFall ❷ Property

Syntax `AP.S2Dio.OutRiseFall`

Data Type Boolean

True	Fixed Rise and fall times.
False	Variable Rise and fall times.

Description This command enables or disables variable rise and fall times of the pulse train at the XLR and BNC outputs. The fixed transition time is approximetly 16 nanoseconds.

See Also `AP.S2Dio.OutRiseFallTime`

Example See example for `AP.S2Dio.OutFormat`.

AP.S2Dio.OutRiseFallTime ❷ Property

Syntax `AP.S2Dio.OutRiseFallTime(unit$)`

Data Type Double — The rise and fall times may be varied from 16 to 400 nanoseconds.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: sec

Description This command sets the variable rise and fall times of the pulse train at the XLR and BNC outputs.

See Also `AP.S2Dio.OutRiseFall`

Example See example for `AP.S2Dio.OutFormat`.

AP.S2Dio.OutSendInvalid ❷ Property

Syntax `AP.S2Dio.OutSendInvalid`

Data Type Boolean

True	Set
False	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. System Two 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.

AP.S2Dio.OutVoltage ❷ Property

Syntax **AP.S2Dio.Voltage(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are 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 AP.S2Dio.VoltageRdg.

AP.S2Dio.RateRdg ❷ Property

Syntax **AP.S2Dio.RateRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: Hz

Description This command returns a settled reading for the Digital Input/Output Sample Rate meter and zeros the ready count.

See Also AP.S2Dio.RateReady, AP.S2Dio.RateSettling, AP.S2Dio.RateTrig

Example

```
'#Uses "Dioconst.apb"

Sub Main
   Dim rdgA As Double

   'S2Dio Sample Rate meter sample code
   AP.S2Dio.InFormat = XLR_BAL   'XLR balanced input
   AP.S2Dio.InImpedance = Z110   'High impedance input
   AP.S2Dio.RateSettling 5.0, 100e-3, "Hz", 1, 0.0, NONE
```

```
   AP.S2Dio.RateTrig       'Trigger sample rate meter
   Do Until AP.S2Dio.RateReady
      'perform other actions while waiting for readings
      '...
   Loop
   rdgA = AP.S2Dio.RateRdg("Hz") 'Get sample rate
   AP.Prompt.Text = "Ch A = " & rdgA & " Vpp"
   AP.Prompt.ShowWithContinue
   Stop
End Sub
```

AP.S2Dio.RateReady

❷ Property

Syntax `AP.S2Dio.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 `AP.S2Dio.RateRdg` or `AP.S2Dio.RateTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2Dio.RateRdg` command will be guaranteed to return quickly.

See Also `AP.S2Dio.RateRdg, AP.S2Dio.RateSettling, AP.S2Dio.RateTrig`

Example See example for `AP.S2Dio.RateRdg`.

AP.S2Dio.RateSettling ❷ Method

Syntax **AP.S2Dio.RateSettling(***tolerance#, floor#, floorunit$, points%, delay#, algorithm%***)**

Description This command sets the settling parameters for the AP.S2Dio.RateRdg command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also AP.S2Dio.RateRdg, AP.S2Dio.RateReady, AP.S2Dio.RateTrig

Example See example for AP.S2Dio.RateRdg.

AP.S2Dio.RateTrig ❷ Method

Syntax **AP.S2DSP.RateTrig**

Description Causes a restart of the reading cycle and zeros the ready count for the AP.S2Dio.RateRdg command. The reading in progress is aborted.

See Also AP.S2Dio.RateRdg, AP.S2Dio.RateReady, AP.S2Dio.RateSettling

Example See example for AP.S2Dio.RateRdg.

AP.S2Dio.RefRate ❷ Property

Syntax **AP.S2Dio.RefRate(***unit$***)**

Data Type Double — Set Reference Sample Rate value.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: Hz

Description This command sets the Digital Input/Output Sample Rate reference.

See Also AP.S2Dsp.InScaleFreq, AP.S2Dsp.OutRate, AP.S2Dsp.RateRdg

AP.S2Dio.VoltageRdg ❷ Property

Syntax **AP.S2Dio.VoltageRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following unit is valid for this command: Vpp

Description This command returns a settled reading for the Digital Input/Output Voltage meter and zeros the ready count.

See Also AP.S2Dio.VoltageReady, AP.S2Dio.VoltageSettling, AP.S2Dio.VoltageTrig

Example

```
'#Uses "Dioconst.apb"

Sub Main
   Dim rdgA As Double

   'S2Dio Voltage out sample code
   AP.S2Dio.OutFormat = XLR_BAL
   AP.S2Dio.InFormat = XLR_BAL      'XLR balanced input
   AP.S2Dio.VoltageSettling 5.0, 100e-3, VPP, 1, 0.0, _
      NONE
   AP.S2Dio.OutVoltage(VPP) = 1.0   'Set output voltage
   AP.S2Dio.VoltageTrig       'Trigger voltage meter
   Do Until AP.S2Dio.VoltageReady
      'perform other actions while waiting for readings
      '...
   Loop
   rdgA = AP.S2Dio.VoltageRdg(VPP) 'Get voltage reading
   AP.Prompt.Text = "Ch A = " & rdgA & " Vpp"
   AP.Prompt.ShowWithContinue
   Stop
End Sub
```

AP.S2Dio.VoltageReady ❷ Property

Syntax `AP.S2Dio.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 `AP.S2Dio.VoltageRdg` or `AP.S2Dio.VoltageTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2Dio.VoltageRdg` command will be guaranteed to return quickly.

See Also `AP.S2Dio.VoltageRdg, AP.S2Dio.VoltageSettling, AP.S2Dio.VoltageTrig`

Example See example for `AP.S2Dio.VoltageRdg`.

AP.S2Dio.VoltageSettling ❷ Method

Syntax **`AP.S2Dio.VoltageSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.S2Dio.VoltageRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.S2Dio.VoltageRdg, AP.S2Dio.VoltageReady, AP.S2Dio.VoltageTrig`

Example See example for `AP.S2Dio.VoltageRdg`.

AP.S2Dio.VoltageTrig ❷ Method

Syntax `AP.S2DSP.VoltageTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2Dio.DelayRdg` command. The reading in progress is aborted.

See Also `AP.S2Dio.VoltageRdg, AP.S2Dio.VoltageReady, AP.S2Dio.VoltageSettling`

Example See example for `AP.S2Dio.VoltageRdg`.

User Notes

User Notes

User Notes

User Notes

User Notes

DSP Audio Analyzer

AP.S2DSP.Analyzer.ChACoupling ❷ Property

Syntax `AP.S2DSP.Analyzer.ChACoupling`

Data Type Boolean

True	DC coupled.
False	Not DC coupled.

Description This command sets the DSP Audio Analyzer channel A Input Coupling to DC. The level meter can then be used to measure DC.

See Also `AP.S2DSP.Analyzer.ChBCoupling`

Example See example for `AP.S2DSP.Analyzer.ChAFreqRdg`.

AP.S2DSP.Analyzer.ChAFreqRdg ❷ Property

Syntax `AP.S2DSP.Analyzer.ChAFreqRdg(`*unit$*`)`

Data Type Variant

Parameters

Part	Description
unit$	The following units are available: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.

Description This command returns a settled reading for the DSP Audio Analyzer channel A Frequency meter and zeros the ready count.

See Also `AP.S2DSP.Analyzer.ChAFreqReady`, `AP.S2DSP.Analyzer.ChAFreqSettling`, `AP.S2DSP.Analyzer.ChAFreqTrig`

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.Output = True
   AP.Anlr.ChAInput = 2
   AP.Application.PanelOpen apbPanelDSPSmall
```

```
AP.S2Dsp.Program = 1        'Load DSP Audio Analyzer
AP.Application.PanelOpen apbPanelDSPLarge
With AP.S2Dsp.Analyzer
   .ChACoupling = False     'Input AC coupled
   .InputFormat = 1         'Low BW A/D input
   .ChARangeAuto = False    'Fixed Range input
   .ChARange("FFS") = 1.000000 'In Range Full Scale

   .ChALevelSettling 1.000000, 0.000001, "V", 3, _
      0.030000, 1
   .ChAFreqSettling 0.500000, 0.010000, "Hz", 3, _
      0.030000, 1
   Wait .5
   .ChALevelTrig      'Trigger new Level reading
   .ChAFreqTrig       'Trigger new Frequency reading
   Do                 'Wait for new readings
      Loop Until .ChALevelReady And .ChAFreqReady
   var1 = .ChALevelRdg("V")
   var2 = .ChAFreqRdg("Hz")
End With
Text1$= "Channel A Level " & Str$(Format(var1, _
   "##.000")) & "V"
Text2$= "Channel A Frequency " & Str$(Format(var2, _
   "##.000")) & "V"
AP.Prompt.Text = Text1$ & Chr(13) & Text2$ 'Text _
   String and New Line
AP.Prompt.ShowWithContinue          'Display Prompt
Stop
End Sub
```


AP.S2DSP.Analyzer.ChAFreqReady ❷ Property

Syntax **AP.S2DSP.Analyzer.ChAFreqReady**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the DSP Audio Analyzer channel A 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 `AP.S2DSP.Analyzer.ChAFreqRdg` or `AP.S2DSP.Analyzer.ChAFreqTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2DSP.Analyzer.ChAFreqRdg` command will be guaranteed to return quickly.

See Also `AP.S2DSP.Analyzer.ChAFreqRdg`, `AP.S2DSP.Analyzer.ChAFreqSettling`, `AP.S2DSP.Analyzer.ChAFreqTrig`

Example See example for `AP.S2DSP.Analyzer.ChAFreqRdg`.

AP.S2DSP.Analyzer.ChAFreqSettling ❷ Method

Syntax **`AP.S2DSP.Analyzer.ChAFreqSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.S2DSP.Analyzer.ChAFreqRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.S2DSP.Analyzer.ChAFreqRdg`, `AP.S2DSP.Analyzer.ChAFreqReady`, `AP.S2DSP.Analyzer.ChAFreqTrig`

Example See example for `AP.S2DSP.Analyzer.ChAFreqRdg`.

AP.S2DSP.Analyzer.ChAFreqTrig ❷ Method

Syntax `AP.S2DSP.Analyzer.ChAFreqTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.Analyzer.ChAFreqRdg` command. The reading in progress is aborted.

See Also `AP.S2DSP.Analyzer.ChAFreqRdg`, `AP.S2DSP.Analyzer.ChAFreqReady`, `AP.S2DSP.Analyzer.ChAFreqSettling`

Example See example for `AP.S2DSP.Analyzer.ChAFreqRdg`.

AP.S2DSP.Analyzer.ChALevelRdg ❷ Property

Syntax `AP.S2DSP.Analyzer.ChALevelRdg(unit$)`

Data Type Variant

Description This command returns a settled reading for the DSP Audio Analyzer channel A level meter and zeros the ready count.

Parameters

Name	Description
unit$	The following units are available for Digital Input: FFS, %FS, dBFS, Bits, V, Vp, Vpp, dBu, dBV, dBr 1, dBr 2. The following units are available for Analog (Low BW A/D) inputs: V, dBV, dBu, dBr A, dBr B, dBm.

See Also `AP.S2DSP.Analyzer.ChALevelReady`, `AP.S2DSP.Analyzer.ChALevelSettling`, `AP.S2DSP.Analyzer.ChALevelTrig`

Example See example for `AP.S2DSP.Analyzer.ChAFreqRdg`.

AP.S2DSP.Analyzer.ChALevelReady ❷ Property

Syntax `AP.S2DSP.Analyzer.ChALevelReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the DSP Audio Analyzer channel A 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 `AP.S2DSP.Analyzer.ChALevelRdg` or `AP.S2DSP.Analyzer.ChALevelTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2DSP.Analyzer.ChALevelRdg` command will be guaranteed to return quickly.

See Also `AP.S2DSP.Analyzer.ChALevelRdg`, `AP.S2DSP.Analyzer.ChALevelSettling`, `AP.S2DSP.Analyzer.ChALevelTrig`

Example See example for `AP.S2DSP.Analyzer.ChAFreqRdg`.

AP.S2DSP.Analyzer.ChALevelSettling ❷ Method

Syntax **`AP.S2DSP.Analyzer.ChALevelSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.S2DSP.Analyzer.ChALevelRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.S2DSP.Analyzer.ChALevelRdg`, `AP.S2DSP.Analyzer.ChALevelReady`, `AP.S2DSP.Analyzer.ChALevelTrig`

Example See example for `AP.S2DSP.Analyzer.ChAFreqRdg`.

AP.S2DSP.Analyzer.ChALevelTrig ❷ Method

Syntax `AP.S2DSP.Analyzer.ChALevelTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.Analyzer.ChALevelRdg` command. The reading in progress is aborted.

See Also `AP.S2DSP.Analyzer.ChALevelRdg`, `AP.S2DSP.Analyzer.ChALevelSettling`, `AP.S2DSP.Analyzer.ChALevelTrig`

Example See example for `AP.S2DSP.Analyzer.ChAFreqRdg`.

AP.S2DSP.Analyzer.ChARange ❷ Property

Syntax `AP.S2DSP.Analyzer.ChARange(`*`unit$`*`)`

Data Type Double

The following values are the range boundaries for the dBFS 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, -186.639.

If an arbitrary value between the range boundaries is entered the next higher range will be selected.

Parameters

Name	Description
unit$	The following units are available: FFS, %FS, dBFS, dBr 1.

Description This command sets the DSP Audio Analyzer Input Range and returns the nominal full scale of the range in use.

See Also `AP.S2DSP.Analyzer.ChARangeAuto`

Example See example for `AP.S2DSP.Analyzer.ChAFreqRdg`.

AP.S2DSP.Analyzer.ChARangeAuto ❷ Property

Syntax `AP.S2DSP.Analyzer.ChARangeAuto`

Data Type Boolean

True	Auto range
False	Fixed range

Description This command sets the DSP Audio Analyzer channel A 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 `AP.S2DSP.Analyzer.ChARange`

Example See example for `AP.S2DSP.Analyzer.ChAFreqRdg`.

AP.S2DSP.Analyzer.ChBCoupling ❷ Property

Syntax `AP.S2DSP.Analyzer.ChBCoupling`

Data Type Boolean

True	DC coupled.
False	Not DC coupled.

Description This command sets DSP Audio Analyzer channel B Input Coupling to DC. The level meter can then be used to measure DC.

See Also `AP.S2DSP.Analyzer.ChACoupling`

Example See example for `AP.S2DSP.Analyzer.ChBFreqRdg`.

AP.S2DSP.Analyzer.ChBFreqRdg ❷ Property

Syntax `AP.S2DSP.Analyzer.ChBFreqRdg(`*unit$*`)`

Data Type Variant

Description This command returns a settled reading for the DSP Audio Analyzer channel B Frequency meter and zeros the ready count.

Parameters

Part	Description
unit$	The following units are available: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.

See Also

AP.S2DSP.Analyzer.ChBFreqReady,
AP.S2DSP.Analyzer.FreqBTrig,
AP.S2DSP.Analyzer.ChBFreqSettling

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.Output = True
   AP.Anlr.ChBInput = 2
   AP.Application.PanelOpen apbPanelDSPSmall
   AP.S2Dsp.Program = 1       'Load DSP Audio Analyzer
   AP.Application.PanelOpen apbPanelDSPLarge
   With AP.S2Dsp.Analyzer
      .ChBCoupling = False    'Input AC coupled
      .InputFormat = 1        'Low BW A/D input
      .ChBRangeAuto = False   'Fixed Range input
      .ChBRange("FFS") = 1.000000 'In Range Full Scale

      .ChBLevelSettling 1.000000, 0.000001, "V", 3, _
         0.030000, 1
      .ChBFreqSettling 0.500000, 0.010000, "Hz", 3, _
         0.030000, 1
      Wait .5
      .ChBLevelTrig     'Trigger new Level reading
      .ChBFreqTrig      'Trigger new Frequency reading
      Do                'Wait for new readings
         Loop Until .ChBLevelReady And .ChBFreqReady
      var1 = .ChBLevelRdg("V")
      var2 = .ChBFreqRdg("Hz")
   End With
   Text1$= "Channel B Level " & Str$(Format(var1, _
      "##.000")) & "V"
   Text2$= "Channel B Frequency " & Str$(Format(var2, _
      "##.000")) & "V"
   AP.Prompt.Text = Text1$ & Chr(13) & Text2$ 'Text _
      String and New Line
   AP.Prompt.ShowWithContinue       'Display Prompt
```

```
   Stop
End Sub
```

AP.S2DSP.Analyzer.ChBFreqReady ❷ Property

Syntax `AP.S2DSP.Analyzer.ChBFreqReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the DSP Audio Analyzer Frequency B 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 `AP.S2DSP.Analyzer.ChBFreqRdg` or `AP.S2DSP.Analyzer.FreqBTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2DSP.Analyzer.ChBFreqRdg` command will be guaranteed to return quickly.

See Also `AP.S2DSP.Analyzer.ChBFreqRdg`, `AP.S2DSP.Analyzer.ChBFreqSettling`, `AP.S2DSP.Analyzer.ChBFreqTrig`

Example See example for `AP.S2DSP.Analyzer.ChBFreqRdg`.

AP.S2DSP.Analyzer.ChBFreqSettling ❷ Method

Syntax **`AP.S2DSP.Analyzer.ChBFreqSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.S2DSP.Analyzer.ChBFreqRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.S2DSP.Analyzer.ChBFreqRdg`, `AP.S2DSP.Analyzer.FreqBTrig`, `AP.S2DSP.Analyzer.FreqBReady`

Example See example for `AP.S2DSP.Analyzer.ChBFreqRdg`.

AP.S2DSP.Analyzer.ChBFreqTrig ❷ Method

Syntax **`AP.S2DSP.Analyzer.ChBFreqTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.Analyzer.ChBFreqRdg` command. The reading in progress is aborted.

See Also `AP.S2DSP.Analyzer.ChBFreqRdg`, `AP.S2DSP.Analyzer.ChBFreqReady`, `AP.S2DSP.Analyzer.ChBFreqSettling`

Example See example for `AP.S2DSP.Analyzer.ChBFreqRdg`.

AP.S2DSP.Analyzer.ChBLevelRdg ❷ Property

Syntax **`AP.S2DSP.Analyzer.ChBLevelRdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Name	Description
unit$	The following units are available for Digital Input: FFS, %FS, dBFS, Bits, V, Vp, Vpp, dBu, dBV, dBr 1, dBr 2. The following units are available for Analog (Low BW A/D) inputs: V, dBV, dBu, dBr A, dBr B, dBm.

Description This command returns a settled reading for the DSO Audio Analyzer channel B Level meter and zeros the ready count.

See Also `AP.S2DSP.Analyzer.ChBLevelReady`, `AP.S2DSP.Analyzer.ChBLevelSettling`, `AP.S2DSP.Analyzer.ChBLevelTrig`

Example See example for `AP.S2DSP.Analyzer.ChBFreqRdg`.

AP.S2DSP.Analyzer.ChBLevelReady ❷ Property

Syntax **`AP.S2DSP.Analyzer.ChBLevelReady`**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the DSP Audio Analyzer channel B 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 `AP.S2DSP.Analyzer.ChBLevelRdg` or `AP.S2DSP.Analyzer.ChBLevelTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2DSP.Analyzer.ChBLevelRdg` command will be guaranteed to return quickly.

See Also `AP.S2DSP.Analyzer.ChBLevelRdg`, `AP.S2DSP.Analyzer.ChBLevelSettling`, `AP.S2DSP.Analyzer.ChBLevelTrig`,

Example See example for `AP.S2DSP.Analyzer.ChBFreqRdg`.

AP.S2DSP.Analyzer.ChBLevelSettling ❷ Method

Syntax **`AP.S2DSP.Analyzer.ChBLevelSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.S2DSP.Analyzer.ChBLevelRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.S2DSP.Analyzer.ChBLevelRdg`, `AP.S2DSP.Analyzer.ChBLevelReady`, `AP.S2DSP.Analyzer.ChBLevelTrig`

Example See example for `AP.S2DSP.Analyzer.ChBFreqRdg`.

AP.S2DSP.Analyzer.ChBLevelTrig ❷ Method

Syntax **`AP.S2DSP.Analyzer.ChBLevelTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.Analyzer.ChBLevelRdg` command. The reading in progress is aborted.

See Also `AP.S2DSP.Analyzer.ChBLevelRdg`, `AP.S2DSP.Analyzer.ChBLevelReady`, `AP.S2DSP.Analyzer.ChBLevelSettling`

Example See example for `AP.S2DSP.Analyzer.ChBFreqRdg`.

AP.S2DSP.Analyzer.ChBRange ❷ Property

Syntax **`AP.S2DSP.Analyzer.ChBRange(`*`unit$`*`)`**

Data Type Double

The following values are the range boundaries for the dBFS 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, -186.639.

If an arbitrary value between the range boundaries is entered the next higher range will be selected.

Parameters

Name	Description

	unit$	The following units are available: FFS, %FS, dBFS, dBr 2.

Description This command sets the DSP Audio Analyzer Input Range and returns the nominal full scale of the range in use.

See Also `AP.S2DSP.Analyzer.ChBRangeAuto`

Example See example for `AP.S2DSP.Analyzer.ChBFreqRdg`.

AP.S2DSP.Analyzer.ChBRangeAuto ❷ Property

Syntax **`AP.S2DSP.Analyzer.ChBRangeAuto`**

Data Type Boolean

True	Auto range
False	Fixed range

Description This command sets the DSP Audio Analyzer channel B 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 `AP.S2DSP.Analyzer.ChBRange`

Example See example for `AP.S2DSP.Analyzer.ChBFreqRdg`.

AP.S2DSP.Analyzer.FilterHP ❷ Property

Obsolete Command not recomended for new design.

Syntax **`AP.S2DSP.Analyzer.FilterHP`**

Data Type Integer

0	<10 Hz
1	22 Hz
2	100 Hz
3	400 Hz

Description This command selects the DSP Audio Analyzer function meter High Pass filter.

See Also `AP.S2DSP.Analyzer.FuncFilterHiPass`

Example

```
Sub Main
End Sub
```

Comment The example program produces a graph that displays the frequency response for the 100Hz High Pass filter.

AP.S2DSP.Analyzer.FilterWeighting ❷ Property

Obsolete Command not recomended for new design.

Syntax **`AP.S2DSP.Analyzer.FilterWeighting`**

Data Type Integer

0	UnWeighted: measures with flat frequency response from approximately 5 Hz to 1/2 the present sample rate.
1	"A" Weighting: processes the signal through a psophometric weighting filter meeting the ANSI A weighting specification before measuring the resulting amplitude.
2	CCIR Weighting: processes the signal with a CCIR-468 weighting filter before measurement, and is normally used in conjunction with the 4/sec QPK Reading Rate selection.
3	20kHz lowpass: six-pole lowpass filter.
4	15kHz lowpass: six-pole lowpass filter.
5	"F" Weighting: is based on recent psychoacoustic research.
6	CCITT Weighting:
7	C-Message Weighting:

Description This command selects the DSP Audio Analyzer Filter Weighting for the function meter.

Note: This command has been replaced by the `AP.S2DSP.Analyzer.FuncFilterLP` and `AP.S2DSP.Analyzer.FuncFilter` commands. The `AP.S2DSP.Analyzer.FilterWeighting` command will continue to function as documented here but will be removed from all visible areas within APWIN.

AP.S2DSP.Analyzer.FuncBPBRFreq ❷ Property

Syntax **AP.S2DSP.Analyzer.FuncBPBRFreq(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	The following units are available: Hz, F/R, dHz, %Hz, cent, octs, decs, d%, dPPM.

Description This command sets the DSP Audio Analyzer BandPass/BandReject filter to the frequency value passed.

The DSP-implemented Bandpass filter affects only the main (Reading) meter, not the Level Monitor 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 AP.S2DSP.Analyzer.FuncBPBRTuning, AP.S2DSP.Analyzer.FuncMode

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.Output = True
   AP.Anlr.ChAInput = 2
   AP.Application.PanelOpen apbPanelDSPSmall
   AP.S2Dsp.Program = 1       'Load DSP Audio Analyzer
   AP.Application.PanelOpen apbPanelDSPLarge
   With AP.S2Dsp.Analyzer
      .InputFormat = 1        'Low BW A/D input
      .FuncInput = 0          'Set Ch A radio button
      .FuncMode = 5           'Mode Bandpass
      .RdgRate = 4            'Set Reading Rate 32/Sec
      .FuncDetector = 0       'Set Detector to RMS
      .FuncBPBRTuning = 4     'Fixed Tuning at 1kHz
```

```
      .FuncBPBRFreq("Hz") = 999.999046

      .FuncSettling 3.000000, 0.000000, "V", 3, _
         0.100000, 1
      Wait .5
      .FuncTrig               'Trigger new reading
      Do                      'Wait for new reading
         Loop Until .FuncReady
      var1 =   .FuncRdg("V")  'Return reading
   End With
   Text$= "Bandpass Amplitude " & Str$(Format(var1,
"##.000")) & "V"
   AP.Prompt.Text = Text$ & Chr(13) 'Text and New Line
   AP.Prompt.ShowWithContinue       'Display Prompt
   Stop
End Sub
```

AP.S2DSP.Analyzer.FuncBPBRTuning ❷ Property

Syntax **AP.S2DSP.Analyzer.FuncBPBRTuning**

Data Type Integer

The following list is for System One.

0	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.
1	Sweep Track: the filter tracks the frequency of whichever generator is selected in the Source 1 or Source 2 fields of the Sweep panel.
2	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 System Two's analog output.
3	DGen Track: the filter will automatically track the frequency of the Digital Generator. This mode would normally be used

4	when sweeping digital input-digital output devices with stimulus coming from System Two's Digital Generator. 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 DSP Audio Analyzer Bandpass Bandreject filter Tuning source.

See Also `AP.S2DSP.Analyzer.FuncBPBRFreq`

Example See example for `AP.S2DSP.Analyzer.FuncBPBRFreq`.

AP.S2DSP.Analyzer.FuncBPHarmonic ❷ Property

Obsolete Command not recomended for new design.

Syntax **`AP.S2DSP.Analyzer.FuncBPHarmonic`**

Data Type Integer

0	Fundamental
1	2nd Harmonic
2	3nd Harmonic
3	4nd Harmonic
4	5nd Harmonic

Description This command sets the Digital Domain Audio Analyzer bandpass filter so that it may be automatically tuned to the Fundamental, 2nd, 3rd, 4th, or 5th Harmonic. This permits automatic tracking of the fundamental frequency or a specific harmonic from 2nd through 5th.

Note: This command has been replaced by the `AP.S2DSP.Analyzer.FuncFilter` command. The `AP.S2DSP.Analyzer.BPHarmonic` command will continue to function as documented here but will be removed from all visible areas within APWIN.

AP.S2DSP.Analyzer.FuncDetector

❷ Property

Syntax `AP.S2DSP.Analyzer.FuncDetector`

Data Type Integer

0	RMS.
1	Fast RMS.
2	Qpeak.

Description This command selects the DSP Audio Analyzer Detector Type for function meter.

See Also `AP.S2DSP.Analyzer.FuncInput`, `AP.S2DSP.Analyzer.RdgRate`, `AP.S2DSP.Analyzer.FuncMode`, `AP.S2DSP.Analyzer.FuncRange`, `AP.S2DSP.Analyzer.FuncRangeAuto`

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.Output = True
   AP.Anlr.ChAInput = 2
   AP.Application.PanelOpen apbPanelDSPSmall
   AP.S2Dsp.Program = 1        'Load DSP Audio Analyzer
   AP.Application.PanelOpen apbPanelDSPLarge
   With AP.S2Dsp.Analyzer
      .InputFormat = 1         'Low BW A/D input
      .FuncMode = 3            'THD+N measurement Mode
      .FuncRangeAuto = True    'Auto Range
'     .ChARange("dBFS") = 0.000000 'Use with Fixed Range
      .RdgRateV2 = 0           'Auto Reading Rate
      .FuncDetector = 0        'RMS Detector
      .FilterHP = 0            'Set HP Filter to <10Hz
      .FuncFilterLP = 1        'Set LP Filter to 20kHz
      .FuncFilter = 0          'No Auxiliary Filter

      .FuncSettling 3.000000, 0.000010, "%", 3, _
         0.100000, 1
      Wait .5
```

```
        .FuncTrig                ‘Trigger new Function _
           meter reading
        Do                       ‘Wait for new readings
           Loop Until .FuncReady
        var = .FuncRdg(“%”)      ‘Get Reading
     End With
     Text$= “THD+N = ” & Str$(Format(var, “##.00000")) _
        & ”%"
     AP.Prompt.Text = Text$ & Chr(13) ‘Text and New Line
     AP.Prompt.ShowWithContinue ‘Display Prompt
     Stop
End Sub
```

AP.S2DSP.Analyzer.FuncFilter ❷ Property

Syntax **AP.S2DSP.Analyzer.FuncFilter**

Data Type Integer

The following list contains the selections relevant to the AP.S2DSP.Analyzer.FuncMode command Ampitude, THD+N Abs, THD+N Ratio, and 2-Ch Ratio selections.

0	None
1	“A” Weighting
2	CCIR Weighting
3	F Weighting
4	CCITT Weighting
5	C-Message Weighting

The following list contains the selections relevant to the AP.S2DSP.Analyzer.FuncMode command Bandpass selection.

0	Narrow
1	Narrow, Freq x2
2	Narrow, Freq x3
3	Narrow, Freq x4
4	Narrow, Freq x5

The following list contains the selections relevant to the AP.S2DSP.Analyzer.FuncMode command Crosstalk selection.

0	Narrow

Description This command selects the DSP Audio Analyzer Weighting Filter.

See Also AP.S2DSP.Analyzer.FuncFilterHP, AP.S2DSP.Analyzer.FuncFilterLP

Example See example for AP.S2DSP.Analyzer.FuncDetector.

AP.S2DSP.Analyzer.FuncFilterHP ❷ Property

Syntax **AP.S2DSP.Analyzer.FuncFilterHP**

Data Type Integer

0	<10 Hz
1	22 Hz
2	100 Hz
3	400 Hz

Description This command selects the DSP Audio Analyzer High Pass filter used with the function meter.

See Also AP.S2DSP.Analyzer.FuncFilterLowPass

Example See example for AP.S2DSP.Analyzer.FuncDetector.

AP.S2DSP.Analyzer.FuncFilterLP ❷ Property

Syntax **AP.S2DSP.Analyzer.FuncFilterLP**

Data Type Integer

0	22 kHz
1	30 kHz
2	80 kHz

3	>500 kHz

Description This command selects the DSP Audio Analyzer Low Pass filter used with the function meter.

See Also `AP.S2DSP.Analyzer.FuncFilterHP`

Example See example for `AP.S2DSP.Analyzer.FuncDetector`.

AP.S2DSP.Analyzer.FuncInput ❷ Property

Syntax **`AP.S2DSP.Analyzer.FuncInput`**

Data Type Integer

0	Channel A
1	Channel B

Description This command selects the DSP Audio Analyzer channel A or channel B to be used for measurements with the Function meter.

See Also
`AP.S2DSP.Analyzer.RdgRate`,
`AP.S2DSP.Analyzer.FuncDetector`,
`AP.S2DSP.Analyzer.FuncMode`,
`AP.S2DSP.Analyzer.FuncRange`,
`AP.S2DSP.Analyzer.FuncRangeBAuto`

Example See example for `AP.S2DSP.Analyzer.FuncDetector`.

AP.S2DSP.Analyzer.FuncMode ❷ Property

Syntax **`AP.S2DSP.Analyzer.FuncMode`**

Data Type Integer

The following list is for System One.

0	2-Channel: the main meter connects to the channel selected by the AP.S2DSP.Analyzer.FuncInput command, while the Level monitor connects to the opposite channel. The bandpass/bandreject filter is not used. This function provides

simultaneous readings of level on both stereo channels and thus permits frequency response measurements of digital stereo devices in a single sweep.

1 Ratio: the main meter connects to the channel selected by the AP.S2DSP.Analyzer.FuncInput command, while the Level monitor connects to the opposite channel. The bandpass/bandreject filter is not used. The display of the main meter is not the directly-measured amplitude of the selected channel, but is the amplitude ratio (in x/y or % units) or level difference (in dB units) of the two channels. Ratio function thus provides an interchannel balance or differential gain measurement, or a non-frequency-selective crosstalk measurement. Ratio mode works only when the selected channel amplitude is equal to or less than the amplitude on the opposite channel. If the amplitude on the selected channel is greater than the opposite channel, the display will be clipped at 0 dB and it will be necessary to select the other channel to obtain a reading.

2 Crosstalk: the main meter connects to the channel selected by the AP.S2DSP.Analyzer.FuncInput command, while the Level monitor connects to the opposite channel. The bandpass filter processes the signal before the main meter measures it. The display of the mainmeter is not the directly-measured amplitude of the selected channel, but is the amplitude ratio (in x/y or % units) or level difference (in dB units) of the two channels. Crosstalk function thus provides a frequency-selective crosstalk measurement, permitting accurate crosstalk measurements even when the crosstalk signal is lower in amplitude than the wideband noise level. Crosstalk mode works only when the selected channel amplitude is equal to or less than the amplitude on the opposite channel. If the ampliltude on the selected channel is greater than the opposite channel, the display will be clipped at 0 dB and it will be necessary to select the other channel to obtain a reading.

3 THD+N relative: the main meter and the Level monitor both connect to the channel selected by the A or B radio buttons. The bandreject (notch) filter processes the signal to remove the fundamental signal before the main meter measures it.

The display of the main meter is not the directly-measured amplitude of the notch-filtered channel, but is the amplitude ratio (in x/y or % units) or level difference (in dB units) of the main meter with respect to the Level meter, which is measuring the unfiltered input signal. The result is THD+N relative to the input signal.

4 THD+N absolute: the main meter and the Level monitor both connect to the channel selected by the AP.S2DSP.Analyzer.FuncInput command. The bandreject (notch) filter processes the signal to remove the fundamental signal before the main meter measures it. The display of the main meter is the directly-measured amplitude of the notch-filtered channel and thus does not depend upon the reading of the Level monitor. The available units (obtained by clicking on the down arrow at the right of the main meter display field) are the normal digital domain absolute amplitude units of %FS, FFS, dBFS, or bits. Digital domain THD+N in absolute units is particularly useful when performing amplitude sweeps on a digital I/O device or an A/D converter, where use of relative units obscures the basic noise-limited operation of many devices. An ideal device will exhibit constant THD+N at all amplitudes.

5 Bandpass: the main meter and the Level monitor both connect to the channel selected by the AP.S2DSP.Analyzer.FuncInput command. The bandpass filter processes the signal to remove the fundamental signal before the main meter measures it. The display of the main meter is the directly-measured amplitude of the notch-filtered channel and does not depend upon the reading of the Level monitor. Digital domain bandpass function is useful when measuring very low-amplitude signals, approaching or below the wideband noise level, as is commonly done when making input/output linearity (amplitude) sweeps on a digital I/O device or an A/D converter.

6 Noise:

Description This command selects the analysis mode of the DSP Audio Analyzer Function meter.

See Also `AP.S2DSP.Analyzer.FuncInput`, `AP.S2DSP.Analyzer.FuncReady`, `AP.S2DSP.Analyzer.FuncSettling`, `AP.S2DSP.Analyzer.FuncTrig`, `AP.S2DSP.Analyzer.RdgRate`, `AP.S2DSP.Analyzer.FuncRange`, `AP.S2DSP.Analyzer.FuncRangeAuto`

Example See example for `AP.S2DSP.Analyzer.FuncDetector`.

AP.S2DSP.Analyzer.FuncRange ❷ Property

Syntax **`AP.S2DSP.Analyzer.FuncRange(`*`unit$`*`)`**

Data Type Double

The following values are the range boundaries for the dBFS 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, -186.639.

If an arbitrary value between the range boundaries is entered the next higher range will be selected.

Parameters

Name	Description
`unit$`	The following units are available: FFS, %FS, dBFS, dBr 1, dBr 2.

Description This command sets the DSP Audio Analyzer Function meter Range.

See Also `AP.S2DSP.Analyzer.FuncInput`, `AP.S2DSP.Analyzer.RdgRate`, `AP.S2DSP.Analyzer.FuncMode`, `AP.S2DSP.Analyzer.FuncRangeAuto`

Example See example for `AP.S2DSP.Analyzer.FuncDetector`.

AP.S2DSP.Analyzer.FuncRangeAuto ❷ Property

Syntax **`AP.S2DSP.Analyzer.FuncRangeAuto`**

Data Type Boolean

True	Auto range.
False	Fixed range.

Description This command sets the DSP Audio Analyzer Function meter to Auto or Fixed Range.

See Also `AP.S2DSP.Analyzer.FuncInput`, `AP.S2DSP.Analyzer.RdgRate`, `AP.S2DSP.Analyzer.FuncMode`, `AP.S2DSP.Analyzer.FuncRange`

Example See example for `AP.S2DSP.Analyzer.FuncDetector`.

AP.S2DSP.Analyzer.FuncRdg ❷ Property

Syntax **`AP.S2DSP.Analyzer.FuncRdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description
unit$	The following units V, dBu, dBV, dBr, dBg, dBm, W are available for the following Function meter Modes: Amplitude, Bandpass, and THD+N Amplitude. The following units %, dB, X/Y are available for the following Function meter Modes: THD+N Ratio, 2-Ch Ratio, and Crosstalk.

Description This command returns a settled reading from the DSP Audio Analyzer Function meter and zeros the ready count.

See Also `AP.S2DSP.Analyzer.FuncInput`, `AP.S2DSP.Analyzer.FuncMode`, `AP.S2DSP.Analyzer.FuncReady`, `AP.S2DSP.Analyzer.FuncSettling`, `AP.S2DSP.Analyzer.FuncTrig`

Example See example for `AP.S2DSP.Analyzer.FuncDetector`.

AP.S2DSP.Analyzer.FuncReady ❷ Property

Syntax `AP.S2DSP.Analyzer.FuncReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the DSP 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 `AP.S2DSP.Analyzer.FuncRdg` or `AP.S2DSP.Analyzer.FuncTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2DSP.Analyzer.FuncRdg` command will be guaranteed to return quickly.

See Also `AP.S2DSP.Analyzer.FuncInput`, `AP.S2DSP.Analyzer.FuncRdg`, `AP.S2DSP.Analyzer.FuncSettling`, `AP.S2DSP.Analyzer.FuncTrig`

Example See example for `AP.S2DSP.Analyzer.FuncDetector`.

AP.S2DSP.Analyzer.FuncSettling ❷ Method

Syntax **`AP.S2DSP.Analyzer.FuncSettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.S2DSP.Analyzer.FuncRdg` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.S2DSP.Analyzer.FuncInput`,
`AP.S2DSP.Analyzer.FuncRdg`,
`AP.S2DSP.Analyzer.FuncReady`,
`AP.S2DSP.Analyzer.FuncTrig`

Example See example for `AP.S2DSP.Analyzer.FuncDetector`.

AP.S2DSP.Analyzer.FuncTrig ❷ Method

Syntax **`AP.S2DSP.Analyzer.FuncTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.Analyzer.FuncRdg` command. The reading in progress is aborted.

See Also `AP.S2DSP.Analyzer.FuncInput`,
`AP.S2DSP.Analyzer.FuncRdg`,
`AP.S2DSP.Analyzer.FuncReady`,
`AP.S2DSP.Analyzer.FuncSettling`

Example See example for `AP.S2DSP.Analyzer.FuncDetector`.

AP.S2DSP.Analyzer.InputFormat ❷ Property

Syntax **`AP.S2DSP.Analyzer.InputFormat`**

Data Type Integer

0	Digital: To view and measure digital domain signals.
1	Low BW (1x) A/D: To measure analog domain signals up to 24 kHz with maximum dynamic range and frequency resolution, select Low BW (1x) A/D. The 1x indicates that these A/D converters operate directly at the Internal Sample Rate.
2	Low BW (/4) A/D: To measure analog domain signals up to 6kHz with greater frequency resolution, select Low BW (/4) A/D. The /4 indicates that these A/D converters operate at the Internal Sample Rate divided by four.

Description This command sets the DSP Audio Analyzer Input Format.

Example See example for `AP.S2DSP.Analyzer.FuncDetector`.

AP.S2DSP.Analyzer.RdgRate ❷ Property

Obsolete Command not recomended for new design.

Syntax **`AP.S2DSP.Analyzer.RdgRate`**

Data Type Integer

0	Auto RMS: 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.
1	4/Sec RMS
2	8/Sec RMS
3	16/Sec RMS
4	32/Sec RMS
5	64/Sec RMS
6	4/Sec QPK

Description This command sets the Digital Domain Audio Analyzer detector response and controls the update rate (integration period) of all three meters of this program. Selections include Auto with RMS detection, fixed rates from 4/sec through 64/sec with the RMS detector, and a 4/sec rate with a quasi-peak detector which complies with CCIR recommendation 468-4 and earlier.

Note: This command has been replaced by the AP.S2DSP.Analyzer.RdgRateV2 command. The AP.S2DSP.Analyzer.RdgRate command will continue to function as documented here but will be removed from all visible areas within APWIN.

AP.S2DSP.Analyzer.RdgRateV2

❷ Property

Syntax `AP.S2DSP.Analyzer.RdgRateV2`

Data Type Integer

0	Auto RMS: 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.
1	4/Sec
2	8/Sec
3	16/Se
4	32/Sec
5	64/Sec
6	128/Sec

Description This command sets the DSP Audio Analyzer Reading update Rate (integration period) for all of the meters in this DSP program.

See Also `AP.S2DSP.Analyzer.FuncInput`, `AP.S2DSP.Analyzer.FuncDetector`, `AP.S2DSP.Analyzer.FuncMode`, `AP.S2DSP.Analyzer.FuncRange`, `AP.S2DSP.Analyzer.FuncRangeAuto`

Example See example for `AP.S2DSP.Analyzer.FuncDetector`.

User Notes

User Notes

User Notes

User Notes

Digital Data Analyzer

AP.S2DSP.Bittest.ChADataRdg ❷ Property

Syntax **AP.S2DSP.Bittest.ChADataRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: dec.

Description This command returns a unsettled reading for the Digital Data Analyzer channel A Data meter and zeros the ready count.

See Also AP.S2DSP.Bittest.ChADataReady, AP.S2DSP.Bittest.ChADataTrig

Example

```
Sub Main
   AP.Application.NewTest
   AP.S2Dsp.Program = 6
   AP.DGen.Wfm 4, 5
   AP.DGen.ChAAmpl("dBFS") = -3
   AP.DGen.OutDitherType = 3
   AP.DGen.Output = True
   AP.S2Dio.InFormat = 3
   AP.S2Dsp.BitTest.ChADataTrig  'Trigger a new reading
   Do
      Ready = AP.S2Dsp.BitTest.ChADataReady
   Loop Until Ready > 0          'Wait for new reading
   Reading1 = AP.S2Dsp.BitTest.ChADataRdg("dec")
                                 'Get new reading
   NewLine$ = Chr(13)
   a$= "Ch A Data "+Left(Str$(Reading1),8)+" dec"
   AP.Prompt.Text = a$ + NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S2DSP.Bittest.ChADataReady ❷ Property

Syntax `AP.S2DSP.Bittest.ChADataReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Data Analyzer channel A Data 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 `AP.S2DSP.Bittest.ChADataRdg` or `AP.S2DSP.Bittest.ChADataTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.Bittest.ChADataRdg` command will be guaranteed to return quickly.

See Also `AP.S2DSP.Bittest.ChADataRd`, `AP.S2DSP.Bittest.ChADataTrig`

Example Output See example for `AP.S2DSP.Bittest.ChADataRdg`.

AP.S2DSP.Bittest.ChADataTrig ❷ Method

Syntax `AP.S2DSP.Bittest.ChADataTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.Bittest.ChADataRdg` command. The reading in progress is aborted.

See Also `AP.S2DSP.Bittest.ChADataRdg`, `AP.S2DSP.Bittest.ChADataReady`

Example Output See example for `AP.S2DSP.Bittest.ChADataRdg`.

AP.S2DSP.Bittest.ChAErrRdg ❷ Property

Syntax **AP.S2DSP.Bittest.ChAErrRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: dec.

Description This command returns a unsettled reading for the Digital Data Analyzer channel A Errors meter and zeros the ready count.

See Also AP.S2DSP.Bittest.ChAErrReady, AP.S2DSP.Bittest.ChAErrTrig

Example

```
Sub Main
   AP.Application.NewTest
   AP.S2Dsp.Program = 6
   AP.DGen.Wfm 4, 5
   AP.DGen.ChAAmpl("dBFS") = -3
   AP.DGen.OutDitherType = 0
   AP.DGen.Output = True
   AP.S2Dio.InFormat = 3
   With AP.S2Dsp.BitTest
      .DisplayError = 0       'Error display normal
      .RdgRate = 3            'Reading rate to 16/second
      .Wfm = 4 'Set waveform analysis pattern to constant
      .FreezeOnError = False  'Don't freeze data on error
      .ChAErrTrig             'Trigger a new reading
      Do
         Ready = .ChAErrReady
      Loop Until Ready > 0    'Wait for new reading
      Reading1 = .ChAErrRdg("dec")  'Get new reading
   End With
   NewLine$ = Chr(13)
   a$= "Ch A Errors "+Left(Str$(Reading1),8)+" dec"
   AP.Prompt.Text = a$ + NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S2DSP.Bittest.ChAErrReady ❷ Property

Syntax `AP.S2DSP.Bittest.ChAErrReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Data Analyzer channel A Errors 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 `AP.S2DSP.Bittest.ChAErrRdg` or `AP.S2DSP.Bittest.ChAErrTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S1DSP.Bittest.ChAErrRdg` command will be guaranteed to return quickly.

See Also `AP.S2DSP.Bittest.ChAErrRdg`, `AP.S2DSP.Bittest.ChAErrTrig`

Example See example for `AP.S2DSP.Bittest.ChAErrRdg`.

AP.S2DSP.Bittest.ChAErrTrig ❷ Method

Syntax `AP.S2DSP.Bittest.ChAErrTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.Bittest.ChAErrRdg` command. The reading in progress is aborted.

See Also `AP.S2DSP.Bittest.ChAErrRdg`, `AP.S2DSP.Bittest.ChAErrReady`

Example See example for `AP.S2DSP.Bittest.ChAErrRdg`.

AP.S2DSP.Bittest.ChBDataRdg ❷ Property

Syntax **AP.S2DSP.Bittest.ChBDataRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: dec.

Description This command returns a unsettled reading for the Digital Data Analyzer channel B Data meter and zeros the ready count.

See Also AP.S2DSP.Bittest.ChBDataReady, AP.S2DSP.Bittest.ChBDataTrig

Example

```
Sub Main
   AP.Application.NewTest
   AP.S2Dsp.Program = 6
   AP.DGen.Wfm 4, 5
   AP.DGen.ChAAmpl("dBFS") = -3
   AP.DGen.OutDitherType = 3
   AP.DGen.Output = True
   AP.S2Dio.InFormat = 3
   AP.S2Dsp.BitTest.ChBDataTrig  'Trigger a new reading
   Do
      Ready = AP.S2Dsp.BitTest.ChBDataReady
   Loop Until Ready > 0          'Wait for new reading
   Reading1 = AP.S2Dsp.BitTest.ChBDataRdg("dec")
                                 'Get new reading
   NewLine$ = Chr(13)
   a$= "Ch B Data "+Left(Str$(Reading1),8)+" dec"
   AP.Prompt.Text = a$ + NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S2DSP.Bittest.ChBDataReady ❷ Property

Syntax `AP.S2DSP.Bittest.ChBDataReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Data Analyzer channel B Data 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 `AP.S2DSP.Bittest.ChBDataRdg` or `AP.S2DSP.Bittest.ChBDataTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2DSP.Bittest.ChBDataRdg` command will be guaranteed to return quickly.

See Also `AP.S2DSP.Bittest.ChBDataRdg`, `AP.S2DSP.Bittest.ChBDataTrig`

Example See example for `AP.S2DSP.Bittest.ChBDataRdg`.

AP.S2DSP.Bittest.ChBDataTrig ❷ Method

Syntax `AP.S2DSP.Bittest.ChBDataTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.Bittest.ChBDataRdg` comand. The reading in progress is aborted.

See Also `AP.S2DSP.Bittest.ChBDataRdg`, `AP.S2DSP.Bittest.ChBDataReady`

Example See example for `AP.S2DSP.Bittest.ChBDataRdg`.

AP.S2DSP.Bittest.ChBErrRdg ❷ Property

Syntax **AP.S2DSP.Bittest.ChBErrRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	String that designates the desired unit. The following units are valid for this command: dec.

Description This command returns a unsettled reading for the Digital Data Analyzer channel B Errors meter and zeros the ready count.

See Also AP.S2DSP.Bittest.ChBErrReady, AP.S2DSP.Bittest.ChBErrTrig

Example

```
Sub Main
   AP.Application.NewTest
   AP.S2Dsp.Program = 6
   AP.DGen.Wfm 4, 5
   AP.DGen.ChAAmpl("dBFS") = -3
   AP.DGen.OutDitherType = 0
   AP.DGen.Output = True
   AP.S2Dio.InFormat = 3
   AP.S2Dsp.BitTest.ChBErrTrig   'Trigger a new reading
   Do
      Ready = AP.S2Dsp.BitTest.ChBErrReady
   Loop Until Ready > 0          'Wait for new reading
   Reading1 = AP.S2Dsp.BitTest.ChBErrRdg("dec")
                                 'Get new reading
   NewLine$ = Chr(13)
   a$= "Ch B Errors "+Left(Str$(Reading1),8)+" dec"
   AP.Prompt.Text = a$ + NewLine$
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S2DSP.Bittest.ChBErrReady ❷ Property

Syntax `AP.S2DSP.Bittest.ChBErrReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Digital Data Analyzer channel B Errors 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 `AP.S2DSP.Bittest.ChBErrRdg` or `AP.S2DSP.Bittest.ChBErrTrig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2DSP.Bittest.ChBErrRdg` command will be guaranteed to return quickly.

See Also `AP.S2DSP.Bittest.ChBErrRdg`, `AP.S2DSP.Bittest.ChBErrTrig`

Example See example for `AP.S2DSP.Bittest.ChBErrRdg`.

AP.S2DSP.Bittest.ChBErrTrig ❷ Method

Syntax `AP.S2DSP.Bittest.ChBErrTTrig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.Bittest.ChBErrRdg` command. The reading in progress is aborted.

See Also `AP.S2DSP.Bittest.ChBErrTRdg`, `AP.S2DSP.Bittest.ChBErrTReady`

Example See example for `AP.S2DSP.Bittest.ChBErrRdg`.

AP.S2DSP.Bittest.DisplayError ❷ Property

Syntax `AP.S2DSP.Bittest.DisplayError`

Data Type Integer

0	Normal
1	Maximum.
2	Totalize.

Description This command sets the mode for the Digital Data Analyzer channel A and B Error displays.

Received data is also measured to determine if it matches the data transmitted. Only the number of bits selected in the Resolution field `AP.S2Dio.Resolution` of the Digital I/O panel will be analyzed. This comparison is done with algorithms which are insensitive to delay between the send and receive sections. The number of errors in the received data per measurement interval are counted for each channel. The `AP.S2DSP.Bittest.DisplayError` command selects the type of analysis to be performed. In the Normal mode, the number of errors detected during the last measurement interval are displayed directly in the Ch 1 and Ch 2 Errors fields of the panel. If Error Display is selected as Maximum, the maximum error count during any measurement interval will be held in the display. A running total of all errors may be accumulated by using the Totalize mode of the Error Display field.

See Also `AP.S2DSP.Bittest.RdgRate`

Example See example for `AP.S2DSP.Bittest.ChAErrRdg`.

AP.S2DSP.Bittest.FreezeOnError ❷ Property

Syntax `AP.S2DSP.Bittest.FreezeOnError`

Data Type Boolean

True	Hold first error reading..
False	Continue updating data readings.

Description This command sets or clears the Freeze Data on Error field on the Digital Data Analyzer.

If the `AP.S2DSP.Bittest.FreezeOnError` command is set to (True), the Data fields will continue to display the value which was received when the first error occurred. If `AP.S2DSP.Bittest.FreezeOnError` is set to (False), the Data fields will continue updating, independent of any errors detected.

See Also `AP.S2DSP.Bittest.RdgRate`

Example See example for `AP.S2DSP.Bittest.ChAErrRdg`.

AP.S2DSP.Bittest.RdgRate ❷ Property

Syntax `AP.S2DSP.Bittest.RdgRate`

Data Type Integer

0	Auto reading rate. The reading rate is automatically selected based on the measured frequency.
1	4/ second update rate.
2	8/ second update rate.
3	16/ second update rate.

Description This command sets the rate a which the Data (and Errors) readings are updated.

Example See example for `AP.S2DSP.Bittest.ChAErrRdg`.

AP.S2DSP.BitTest.Wfm

❷ Property

Syntax `AP.S2DSP.BitTest.Wfm`

Data Type Integer

0	Constant
1	Random.
2	Walking-1.
3	Walking-0.
4	Sine.

Description This command selects the Digital Data Analyzer Waveform pattern to analyze.

Example See example for AP.S2DSP.BitTest.ChAErrRdg.

User Notes

User Notes

User Notes

User Notes

User Notes

Multitone Audio Analyzer

AP.S2DSP.FastTest.Ch1Rdg ❷ Property

Syntax **AP.S1DSP.FastTest.Ch1Rdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	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 channel 1 Peak Monitor meter and zeros the ready count.

See Also AP.S2DSP.FastTest.Ch1Ready, AP.S2DSP.FastTest.Ch1Trig

Example

```
Sub Main
   AP.File.OpenTest “FASTTSTC.at2"  ‘Opens test
   Wait 1
   With AP.S2Dsp.FastTest
      .Ch1Trig                    ‘Trigger a new reading
      Do
         Ready1 = .Ch1Ready
         Loop Until Ready1 > 0    ‘Wait for reading
      Reading1 = .Ch1Rdg(“dBFS”) ‘Get reading
   End With
   NewLine$ = Chr(13)
   a$= “Ch1 Peak Mon ” & Left(Str$(Reading1),6) & “dBFS”
   AP.Prompt.Text = a$ & NewLine$ & b$ + NewLine
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S2DSP.FastTest.Ch1Ready ❷ Property

Syntax `AP.S2DSP.FastTest.Ch1Ready`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Multitone Audio Analyzer channel 1 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 `AP.S2DSP.FastTest.Ch1Rdg` or `AP.S2DSP.FastTest.Ch1Trig` commands will zero the ready count.

If the reading is found to be ready, a call to the `AP.S2DSP.FastTest.Ch1Rdg` command will be guaranteed to return quickly.

See Also `AP.S2DSP.FastTest.Ch1Rd, AP.S2DSP.FastTest.Ch1Trig`

Example See example for `AP.S2DSP.FastTest.Ch1Rdg`.

AP.S2DSP.FastTest.Ch1Source ❷ Property

Syntax `AP.S2DSP.FastTest.Ch1Source`

Data Type Integer

The following list contains the selections relevant to the `AP.S2DSP.FastTest.InputFormat` command Digital input selection.

0	A
1	B
2	None

The following list contains the selections relevant to the `AP.S2DSP.FastTest.InputFormat` command Low BW (1x) A/D and High BW (4x) A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Ch. A Generator
5	Ch. B Generator
6	Jitter Signal
7	None

Description This command sets the Multitone Audio Analyzer Channnel 1 Input.

Example See example for `AP.S2DSP.FastTest.FFTLength`.

AP.S2DSP.FastTest.Ch1Trig ❷ Method

Syntax **`AP.S2DSP.FastTest.Ch1Trig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S1DSP.FastTest.Ch1Rdg` command. The reading in progress is aborted.

See Also `AP.S2DSP.FastTest.Ch1Rdg`, `AP.S2DSP.FastTest.Ch1Ready`

Example See example for `AP.S2DSP.FastTest.Ch1Rdg`.

AP.S2DSP.FastTest.Ch2Rdg ❷ Property

Syntax **`AP.S2DSP.FastTest.Ch2Rdg(`*`unit$`*`)`**

Data Type Variant

Parameters

Part	Description

unit$	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 channel 2 Peak Monitor meter and zeros the ready count.

See Also AP.S2DSP.FastTest.Ch2Ready, AP.S2DSP.FastTest.Ch2Trig

Example

```
Sub Main
   AP.File.OpenTest "FASTTSTC.at2"  'Open test
   Wait 1
   With AP.S2Dsp.FastTest
      .Ch2Trig                    'Trigger a new reading
      Do
         Ready2 = .Ch2Ready
         Loop Until Ready2 > 0    'Wait for new reading
      Reading2 = .Ch2Rdg("dBFS") 'Get new reading
   End With
   NewLine$ = Chr(13)
   a$= "Ch2 Peak Mon " & Left(Str$(Reading2),6) & "dBFS"
   AP.Prompt.Text = a$ & NewLine$ & b$ + NewLine
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```


AP.S2DSP.FastTest.Ch2Ready ❷ Property

Syntax AP.S2DSP.FastTest.Ch2Ready

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Multitone Audio Analyzer channel 2 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 AP.S2DSP.FastTest.Ch2Rdg or AP.S2DSP.FastTest.Ch2Trig commands will zero the ready count.

If the reading is found to be ready, a call to the AP.S2DSP.FastTest.Ch2Rdg command will be guaranteed to return quickly.

See Also AP.S2DSP.FastTest.Ch2Rdg, AP.S2DSP.FastTest.Ch2Trig

Example See example for AP.S2DSP.FastTest.Ch2Rdg.

AP.S2DSP.FastTest.Ch2Source ❷ Property

Syntax **AP.S1DSP.FastTest.Ch2Source**

Data Type Integer

The following list contains the selections relevant to the AP.S2DSP.FastTest.InputFormat command Digital input selection.

0	A
1	B
2	None

The following list contains the selections relevant to the AP.S2DSP.FastTest.InputFormat command Low BW (1x) A/D and High BW (4x) A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Ch. A Generator
5	Ch. B Generator
6	Jitter Signal
7	None

Description This command sets the Multitone Audio Analyzer Channnel 2 Input.

Example See example for `AP.S2DSP.FastTest.FFTLength`.

AP.S2DSP.FastTest.Ch2Trig ❷ Method

Syntax **`AP.S2DSP.FastTest.Ch2Trig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.FastTest.Ch2Rdg` comand. The reading in progress is aborted.

See Also `AP.S1DSP.FastTest.Ch2Rdg`, `AP.S1DSP.FastTest.Ch2Ready`

Example See example for `AP.S1DSP.FastTest.Ch2Rdg`.

AP.S2DSP.FastTest.FFTLength ❷ Property

Syntax **`AP.S2DSP.FastTest.FFTLength`**

Data Type Integer

0	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.
1	512
2	1024
3	2048
4	4096
5	8192
6	16384

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 `AP.Sweep.Start` is initiated to acquire and transform, or the F6 or `Ap.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
   AP.File.OpenTest "FasttstB.at2"  'Open test
   With AP.S2Dsp.FastTest
      .InputFormat = 1  'Set input to Low BW A/D
      .Ch1Source = 0 'Set Source to Anlr-A
      .Ch2Source = 1 'Set Source to Anlr-B
      .Mode = 0   'Set Measurement to Spectrum
      .FreqRes("%") = 1 'Set Freq Res to 1%
      .FFTLength = 6 'Set FFT lenght to 16384
      .Processing = 0   'Set Processing to Synchronous
      .TrigSource = 0   'Set Triggering to DGEN
      .TrigDelay("sec") = 0   'Set Trig Delay to 0
      .PhaseDisplay = 0 'Set Ch 2 Phase Display to _
         Independent

      AP.Sweep.Start
      'Attach sweep file
      AP.Sweep.Source1.Table("Fasttst.ads", 0)
      .Mode = 1   'Set Measurement to Response
      AP.Sweep.Reprocess
      .Mode = 2   'Set Measurement to Distortion
      AP.Sweep.Reprocess
      .Mode = 3   'Set Measurement to Noise
      AP.Sweep.Reprocess
      .Mode = 4   'Set Measurement to Masking Curve
      AP.Sweep.Reprocess
   End With
End Sub
```

AP.S2DSP.FastTest.FreqRes ❷ Property

Syntax `AP.S2DSP.FastTest.FreqRes(`*`unit$`*`)`

Data Type Double — Valid amplitude settings are from +/- 0.0 to 13.0 %.

Parameters

Name	Description
unit$	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 `AP.S2DSP.FastTest.FFTLength`.

AP.S2DSP.FastTest.InputFormat ❷ Property

Syntax `AP.S2DSP.FastTest.InputFormat`

Data Type Integer

0	Digital: To view and measure digital domain signals.
1	Low BW (1x) A/D: To measure analog domain signals up to 24 kHz with maximum dynamic range and frequency resolution, select Low BW (1x) A/D. The 1x indicates that these A/D converters operate directly at the Internal Sample Rate.
2	Low BW (/4) A/D: To measure analog domain signals up to 6kHz with greater frequency resolution, select Low BW (/4) A/D. The /4 indicates that these A/D converters operate at the Internal Sample Rate divided by four.

Description This command sets the Multitone Audio Analyzer Input Format.

Example See example for `AP.S2DSP.FastTest.FFTLength`.

AP.S2DSP.FastTest.Mode ❷ Property

Syntax `AP.S2DSP.FastTest.Mode`

Data Type Integer

0	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.
1	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.

2 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.

3 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.

4 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.

5 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 `AP.S2DSP.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 `AP.S2DSP.FastTest.FFTLength`.

AP.S2DSP.FastTest.PhaseDisplay ❷ Property

Syntax **`AP.S2DSP.FastTest.PhaseDisplay`**

Data Type Integer

0	Independent
1	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 `AP.S2DSP.FastTest.FFTLength`.

AP.S2DSP.FastTest.Processing ❷ Property

Syntax **`AP.S2DSP.FastTest.Processing`**

Data Type Integer

0	Synchronous: Normal operation of FASTTEST involves acquisition of a multitone signal which was generated from a multitone waveform file by System Two's Digital Generator. The multitone waveform files furnished with APWIN and System Two 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.
1	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.
2	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 `AP.S2DSP.FastTest.FFTLength`.

AP.S2DSP.FastTest.TrigDelay ❷ Property

Syntax **`AP.S2DSP.FastTest.TrigDelay(`*`unit$`*`)`**

Data Type Double — Values up to 1.365 seconds may be entered.

Parameters

Name	Description
`unit$`	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 filed 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 `AP.S2DSP.FastTest.FFTLength`.

AP.S2DSP.FastTest.TrigSource ❷ Property

Syntax **`AP.S2DSP.FastTest.TrigSource`**

Data Type Integer

`0`	DGen: this selection functions only on Dual Domain units (SYS-2300 series). 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.
`1`	Tight:
`2`	Normal:

3	Loose:
4	External: this selection is operational only with SYS-2300 series (Dual Domain) units. It 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 acquisiton being triggered on the next sample after pin 3 is pulled high.
5	Off: this selection produces untriggered or free-running operation. Acquisition and processing begins as soon as the F9 key, Go, or `AP.Sweep.Start` command is initiated. The Off selection is the recommended triggering mode when System Two 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.

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 `AP.S2DSP.FastTest.FFTLength`.

User Notes

User Notes

User Notes

User Notes

User Notes

Spectrum Analyzer

AP.S2DSP.FFT.Averages ❷ Property

Syntax `AP.S2DSP.FFT.Averages`

Data Type Integer

0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096

Description This command sets the Spectrum Analyzer/Generator 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 AP.S2DSP.FFT.AveragesType

Example

```
Sub Main
   AP.File.OpenTest "FFTTEST1.AT2" 'Open test
   With AP.S2Dsp.FFT
      .InputFormat = 1  'Input for Low BW A/D
      .Ch1Source = 0    'Analyzer A input
      .Ch2Source = 1    'Analyzer B input
      .AverageType = 0  'Averaging to Power _
         (Spectrum Only)
      .Averages = 5     'Number of Averages to 16
      .Length = 6       'FFT Length 16384
      .StartTime("sec") = 0   'Set Start Time to 0 sec
      .SubtractDC = 1   'Subtract Average waveform level
      .WfmDisplay = 0   'Waveform Display to Interpolate
      .Window = 0       'Window to Blackman-Harris
      .TrigDelay("sec") = 0.000000  'Trigger Delay
      .TrigSource = 0   'Trigger Source Free Run
      .TrigSensitivity("dBFS") = -59.999594
      .TrigPolarity = 0 'Trigger Slope Positive
   End With

   AP.Sweep.Start       'Perform FFT
End Sub
```

AP.S2DSP.FFT.AveragesType ❷ Property

Syntax **AP.S2DSP.FFT.AveragesType**

Data Type Integer

The following list contains the selections relevant to the AP.S2DSP.FFT.Window command Blackman-Harris, Hann, Flat-Top, Equiripple, and None Window selections.

0	Power (spectrum only)
1	Sync, re-align
2	Sync

The following list contains the selections relevant to the AP.S2DSP.FFT.Window command None, move to bin center Window selection.

0	Power (spectrum only)
1	Sync, re-align, move center first
2	Sync, re-align, average first
3	Sync, move center first
4	Sync, average first

Description This command sets the type of Averaging the 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 AP.S2DSP.FFT.Averages

Example See example for AP.S2DSP.FFT.Averages.

AP.S2DSP.FFT.Ch1Rdg ❷ Property

Syntax **AP.S2DSP.FFT.Ch1Rdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	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 Spectrum Analyzer channel 1 Peak Monitor meter and zeros the ready count.

See Also AP.S2DSP.FFT.Ch1Ready, AP.S2DSP.FFT.Ch1Trig

Example

```
Sub Main
   AP.File.OpenTest "FFTTEST2.AT2"     'Open test
   AP.S2DSP.FFT.Ch1Source = 1    'Set Ch 1 Source to _
      Anlr-B
   Wait 1
   AP.S2DSP.FFT.Ch1Trig          'Trigger a new reading
   Do
      Ready = AP.S2DSP.FFT.Ch1Ready
   Loop Until Ready > 0          'Wait for new reading
   Reading1 = AP.S2DSP.FFT.Ch1Rdg("FFS")'Get new reading
   NewLine$ = Chr(13)
   a$= "Ch 1 Source "+Left(Str$(Reading1),6)+"FFS"
   AP.Prompt.Text = a$ + NewLine$ + b$ + NewLine
   AP.Prompt.ShowWithContinue
   Beep
   Stop
End Sub
```

AP.S2DSP.FFT.Ch1Ready ❷ Property

Syntax AP.S2DSP.FFT.Ch1Ready

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Spectrum Analyzer channel 1 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 AP.S2DSP.FFT.Ch1Rdg or AP.S2DSP.FFT.Ch1Trig commands will zero the ready count.

If the reading is found to be ready, a call to the AP.S2DSP.FFT.Ch1Rdg command will be guaranteed to return quickly.

See Also AP.S2DSP.FFT.Ch1Rdg, AP.S2DSP.FFT.Ch1Trig

Example See example for AP.S2DSP.FFT.Ch1Rdg.

AP.S2DSP.FFT.Ch1Source ❷ Property

Syntax **AP.S2DSP.FFT.Ch1Source**

Data Type Integer

The following list contains the selections relevant to the AP.S2DSP.FFT.InputFormat command A/D Input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Ch. A Generator
5	Ch. B Generator
6	Jitter Signal
7	None

The following list contains the selections relevant to the AP.S2DSP.FFT.InputFormat command Digital Input selection.

0	A
1	B
2	None

Description This command sets the Spectrum Analyzer Channnel 1 Input.

Example See example for AP.S2DSP.FFT.Ch1Rdg.

AP.S2DSP.FFT.Ch1Trig ❷ Method

Syntax `AP.S2DSP.FFT.Ch1Trig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.FFTGen.Ch1Rdg` command. The reading in progress is aborted.

See Also `AP.S2DSP.FFT.Ch1Rdg, AP.S2DSP.FFT.Ch1Ready`

Example See example for `AP.S2DSP.FFT.Ch1Rdg`.

AP.S2DSP.FFT.Ch2Rdg ❷ Property

Syntax **`AP.S2DSP.FFT.Ch2Rdg(`***`unit$`***`)`**

Data Type Variant

Parameters

Part	Description
unit$	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 Spectrum Analyzer channel 2 Peak Monitor meter and zeros the ready count.

See Also `AP.S2DSP.FFT.Ch2Ready, AP.S2DSP.FFT.Ch2Trig`

Example

```
Sub Main
   AP.File.OpenTest “FFTTEST2.AT2"     ‘Open test
   AP.S2DSP.FFT.Ch2Source = 0    ‘Set Ch 2 Source to _
      Anlr-A
   Wait 1
   AP.S2DSP.FFT.Ch2Trig          ‘Trigger a new reading
   Do
      Ready = AP.S2DSP.FFT.Ch2Ready
   Loop Until Ready > 0       ‘Wait for a new reading
   Reading1 = AP.S2DSP.FFT.Ch2Rdg(“FFS”) ‘Get a new _
      reading
   NewLine$ = Chr(13)
   a$= “Ch 2 Source ”+Left(Str$(Reading1),6)+"FFS"
   AP.Prompt.Text = a$ + NewLine$ + b$ + NewLine
```

```
    AP.Prompt.ShowWithContinue
    Beep
    Stop
End Sub
```

AP.S2DSP.FFT.Ch2Ready ❷ Property

Syntax **AP.S2DSP.FFT.Ch2Ready**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Spectrum Analyzer channel 2 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 AP.S2DSP.FFT.Ch2Rdg or AP.S2DSP.FFT.Ch2Trig commands will zero the ready count.

If the reading is found to be ready, a call to the AP.S2DSP.FFT.Ch2Rdg command will be guaranteed to return quickly.

See Also AP.S2DSP.FFT.Ch2Rdg, AP.S2DSP.FFT.Ch2Trig

Example See example for AP.S2DSP.FFT.Ch2Rdg.

AP.S2DSP.FFT.Ch2Source ❷ Property

Syntax **AP.S2DSP.FFT.Ch2Source**

Data Type Integer

The following list contains the selections relevant to the AP.S2DSP.FFT.InputFormat command A/D input selection.

0	Anlr-A

1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Ch. A Generator
5	Ch. B Generator
6	Jitter Signal
7	None

The following list contains the selections relevant to the `AP.S2DSP.FFT.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Spectrum Analyzer Channnel 2 Input.

Example See example for `AP.S2DSP.FFT.Ch2Rdg`.

AP.S2DSP.FFT.Ch2Trig ❷ Method

Syntax **`AP.S2DSP.FFT.Ch2Trig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.FFT.Ch2Rdg` comand. The reading in progress is aborted.

See Also `AP.S2DSP.FFT.Ch2Rdg, AP.S2DSP.FFT.Ch2Ready`

Example See example for `AP.S2DSP.FFT.Ch2Rdg`.

AP.S2DSP.FFT.InputFormat ❷ Property

Syntax **`AP.S2DSP.FFT.InputFormat`**

Data Type Integer

0	Digital: To view and measure digital domain signals.
1	Low BW (1x) A/D: To measure analog domain signals up to 24 kHz with maximum dynamic range and frequency resolution, select Low BW (1x) A/D. The 1x indicates that these A/D converters operate directly at the Internal Sample Rate.
2	High BW (4x) A/D: To measure analog domain signals up to 80 kHz (with compromises in dynamic range and a 4:1 reduction in maximum frequency resolution relative to the Low Bandwidth A/Ds), select High BW (4x) A/D. The 4x indicates that these converters operate at four times the Internal Sample Rate.
3	Low BW (/4) A/D: To measure analog domain signals up to 6kHz with greater frequency resolution, select Low BW (/4) A/D. The /4 indicates that these A/D converters operate at the Internal Sample Rate divided by four.

Description This command sets the Spectrum Analyzer Input Format.

Example See example for `AP.S2DSP.FFT.Averages`.

AP.S2DSP.FFT.Length ❷ Property

Syntax `AP.S2DSP.FFT.Length`

Data Type Integer

0	256
1	512
2	1024
3	2048
4	4096
5	8192
6	16384
7	256/24k acq.
8	512/24k acq.
9	1024/24k acq.
10	2048/24k acq.

11	4096/24k acq.
12	8192/24k acq.
13	16384/24k acq.

Description This command sets the Spectrum Analyzer acqusition and FFT length.

The first seven selections, without the /24k acq. (0-6), will result in only the specified number of samples being acquired into the acquisitions buffer. Acquisitions will be faster, but there will be no surplus data samples to be analyzed by changing the Start Time.

The last seven selections, with the /24k acq. (7-13), will result in the entire 24 kbyte acquisition buffer being filled even though the transform length is shorter. The additional data may be transformed by changing the Start Time value, or may be examined in the time domain by changing the Source 1 Start and Stop times on the Sweep panel.

Example See example for `AP.S2DSP.FFT.Averages`.

AP.S2DSP.FFT.PreTrig ❷ Property

Obsolete Obsolete command not recomended for new design.

Syntax **`AP.S2DSP.FFT.PreTrig(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
`unit$`	String that designates the desired unit. The following units are valid for this command: sec.

Description This command sets the Spectrum Analyzer Pre-Trigger 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 Pre-Trigger Time 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.

Note: This command has been replaced by the `AP.S2DSP.FFT.TrigDelay` command. The `AP.S2DSP.FFT.PreTrig` command will continue to function as documented here but will be removed from all visible areas within APWIN.

AP.S2DSP.FFT.StartTime

❷ Property

Syntax `AP.S2DSP.FFT.StartTime(`*unit$*`)`

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.

Parameters

Name	Description

unit$	String that designates the desired unit. The following units are valid for this command: sec.

Description This command sets the 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 `AP.S2DSP.FFT.Averages`.

AP.S2DSP.FFT.SubtractAve — Property

Obsolete Obsolete command not recomended for new design.

Syntax **`AP.S2DSP.FFT.SubtractAve`**

Data Type Boolean

True	Subtract average value ON.
False	Subtract average value OFF.

Description 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.

Note: This command has been replaced by the `AP.S2DSP.FFT.SubtractDC` command. The `AP.S2DSP.FFT.SubtractAve` command will continue to function as documented here but will be removed from all visible areas within APWIN.

AP.S2DSP.FFT.SubtractDC ❷ Property

Syntax **`AP.S2DSP.FFT.SubtractDC`**

Data Type Integer

0	DC Coupled
1	Subtract Average
2	Subtract 1/2pk-pk

Description This command sets the Spectrum Analyzer DC offset processing mode.

Example See example for `AP.S2DSP.FFT.Averages`.

AP.S2DSP.FFT.TrigDelay ❷ Property

Syntax **`AP.S2DSP.FFT.TrigDelay(`*`unit$`*`)`**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: sec.

Description This command sets the 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 `AP.S2DSP.FFT.Averages`.

AP.S2DSP.FFT.TrigPolarity ❷ Property

Syntax `AP.S2DSP.FFT.TrigPolarity`

Data Type Integer

0	Positive: time zero will be the first positive-going zero crossing of the trigger signal selected in the Trigger Source field.
1	Negative: time zero will be the first negative-going zero crossing of the selected trigger signal.

Description This command sets the Spectrum Analyzer trigger polarity.

Example See example for `AP.S2DSP.FFT.Averages`.

AP.S2DSP.FFT.TrigSensitivity ❷ Property

Syntax `AP.S2DSP.FFT.TrigSensitivity(unit$)`

Data Type Double — The acceptable range of numbers is between plus and minus 1 for the FFS unit.

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, FS, and dBFS.

Description This command sets the 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 `AP.S2DSP.FFT.Averages`.

AP.S2DSP.FFT.TrigSource ❷ Property

Syntax `AP.S2DSP.FFT.TrigSource`

Data Type Integer

0	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.
1	Ch. 1 Auto:
2	Ch. 1 Fixed:
3	Ch. 2 Auto:
4	Ch. 2 Fixed:
5	External: 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.

6	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.
7	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 System Two. 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.
8	AC Mains: the power line frequency.
9	Jitter Gen: The Digital Input/Output Jitter Generator selection provides a trigger at each positive or negative zero crossing for the selected waveform type.

Description This command sets the Spectrum Analyzer Trigger Source.

The four channel 1 and channel 2 selections are software triggers, monitoring the signal (which may come from Digital or A/D sources) on the specified channel. channel 1 Fix and channel 2 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 1 and 2 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 `AP.S2DSP.FFT.Averages`.

AP.S2DSP.FFT.WfmDisplay ❷ Property

Syntax `AP.S2DSP.FFT.WfmDisplay`

Data Type Integer

0	Interpolate
1	Display Samples
2	Peak Values
3	Absolute Values

Description This command sets the 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, dBm, 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 `AP.S2DSP.FFT.Averages`.

AP.S2DSP.FFT.Window ❷ Property

Syntax **`AP.S2DSP.FFT.Window`**

Data Type Integer

0	Blackman-Harris
1	Hann
2	Flat-Top
3	Equiripple
4	None
5	None, move to bin center

Description This command sets the Spectrum Analyzer Window selection. See Appendix E for FFT Window Discriptions.

See Also `AP.S2DSP.FFT.AveragesType`

Example See example for `AP.S2DSP.FFT.Averages`.

User Notes

User Notes

User Notes

User Notes

User Notes

Digital Interface Analyzer

AP.S2DSP.Intervu.AmplVsTime ❷ Property

Syntax `AP.S2DSP.Intervu.AmplVsTime`

Data Type Integer

0	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.
1	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.
2	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.
3	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
   AP.File.OpenTest "INTERVU1.at2"
   With AP.S2Dsp.Intervu
      .AmplVsTime = 0       'Set to Interpolate
      .AudioMonitor = 0     'Set Audio Monitor
      .JitterDetection = 0 'Set Jitter Detection to_
          Stable Bits
      .TrigSource = 3       'Set Trigger Ch B Transmit _
         Preamble
      .Window = 0           'Set Blackman-Harris Window
   End With
   AP.Sweep.Start
End Sub
```

AP.S2DSP.Intervu.AudioMonitor ❷ Property

Syntax `AP.S2DSP.Intervu.AudioMonitor`

Data Type Integer

0	Audio Monitor: Monitor the imbedded digital audio signal.
1	Jitter Signal: Monitor the demodulated jitter signal.

Description This command determins the audio signal that is sent to the headphone output from the Digital Interface Analyzer.

See Also `AP.Speaker.Mode, AP.Speaker.Source`

Example See example for `AP.S2DSP.Intervu.AmplVsTime`.

AP.S2DSP.Intervu.Averages ❷ Property

Syntax `AP.S2DSP.Intervu.Averages`

Data Type Integer

0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

Description This command sets the Digital Interface Analyzer number of acquisition-and-processing cycles to average.

Example

```
Sub Main
   AP.Application.NewTest
   AP.Application.PanelClose apbPanelAnalogGenSmall
   AP.Application.PanelClose apbPanelAnlrSmall
   AP.DGen.Output = True
   AP.S2Dsp.Program = 3 'Digital Interface Analyzer
   AP.S2Dsp.Intervu.Averages = 4 '16 Averages
   AP.S2Dsp.Intervu.Window = 4   'No Window
   AP.Sweep.Data1.Id = 6055
   AP.Sweep.Source1.Id = 5613
   AP.S2Dio.OutJitterType = 1    'Sine Jitter
   AP.S2Dio.OutJitterAmpl("sec") = 100e-9
   AP.Sweep.Start
   AP.Graph.OptimizeLeft
End Sub
```

AP.S2DSP.Intervu.JitterDetection ❷ Property

Syntax `AP.S2DSP.Intervu.JitterDetection`

Data Type Integer

0	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).
1	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.
2	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.

Description This command determines at which transitions the clock timing is compared to the interface signal..

See Also AP.

Example See example for `AP.S2DSP.Intervu.AmplVsTime`.

AP.S2DSP.Intervu.TrigSource ❷ Property

Syntax **`AP.S2DSP.Intervu.TrigSource`**

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 `AP.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.
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 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.

3 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 `AP.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 Pre-Trigger: operatse in conjunction with 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 acquisiton being triggered on the next sample after pin 3 is pulled high. The External Pre-Trigger selection results in retaining the 256 k samples immediately preceding this sample.
8	External Post-Trigger: operatse in conjunction with 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 acquisiton being triggered on the next sample after pin 3 is pulled high. The External Post-Trigger selection retains the 256 k samples immediately following this sample.

Description This command defines the trigger source that is used to trigger an acqusition.

Example See example for `AP.S2DSP.Intervu.AmplVsTime`.

AP.S2DSP.Intervu.Window ❷ Property

Syntax `AP.S2DSP.Intervu.Window`

Data Type Integer

0	Blackman-Harris

1	Hann
2	Flat-Top
3	Equiripple
4	None

Description This command sets the Digital Interface Analyzer Window selection. See Appendix E for FFT Window Discriptions.

Example See example for `AP.S2DSP.Intervu.AmplVsTime`.

User Notes

User Notes

User Notes

Quasi-Anechoic Acoustical Tester

AP.S2DSP.MLS.Ch1Rdg ❷ Property

Syntax **AP.S2DSP.MLS.Ch1Rdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	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 Quasi-Anechoic Acoustical Tester channel 1 Peak Monitor meter and zeros the ready count.

See Also AP.S2DSP.MLS.Ch1Ready, AP.S2DSP.MLS.Ch1Trig

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.Wfm 7, 0
   AP.Gen.Output = True
   AP.Anlr.ChAInput = 2
   AP.Anlr.ChBInput = 2
   With AP.S2Dsp
      .Program = 5
      .Mls.InputFormat = 1        'Low BW A/D input
      .Mls.Ch1Source = 0          'Analyzer A input
      .Mls.Ch2Source = 1          'Analyzer B input

   Wait 0.5
      .Mls.Ch1Trig                'Trigger Ch 1 reading
      .Mls.Ch2Trig                'Trigger Ch 2 reading
      Do
         Ready1 = .Mls.Ch1Ready   'Check status
         Ready2 = .Mls.Ch2Ready   'Ckeck status
      Loop Until Ready1 > 0 And Ready2 > 0

      Reading1 = .Mls.Ch1Rdg("FFS") 'Get reading
      Reading2 = .Mls.Ch2Rdg("FFS") 'Get reading
```

```
   End With
   NewLine$ = Chr(13)
   a$= “Ch1 Peak Mon ”+Left(Str$(Reading1),6)+" FFS"
   b$= “Ch2 Peak Mon ”+Left(Str$(Reading2),6)+" FFS"
   AP.Prompt.Text = a$ + NewLine$ + b$ + NewLine
   AP.Prompt.ShowWithContinue
   Stop
End Sub
```

AP.S2DSP.MLS.Ch1Ready

❷ Property

Syntax **AP.S2DSP.MLS.Ch1Ready**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Quasi-Anechoic Acoustical Tester channel 1 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 AP.S2DSP.MLS.Ch1Rdg or AP.S2DSP.MLS.Ch1Trig commands will zero the ready count.

If the reading is found to be ready, a call to the AP.S2DSP.MLS.Ch1Rdg command will be guaranteed to return quickly.

See Also AP.S2DSP.MLS.Ch1Rdg, AP.S2DSP.MLS.Ch1Trig

Example See example for AP.S2DSP.MLS.Ch1Rdg.

AP.S2DSP.MLS.Ch1Source

❷ Property

Syntax **AP.S2DSP.MLS.Ch1Source**

Data Type Integer

The following list contains the selections relevant to the `AP.S2DSP.MLS.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Ch. A Generator
5	Ch. B Generator
6	Jitter Signal
7	None

The following list contains the selections relevant to the `AP.S2DSP.MLS.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Quasi-Anechoic Acoustical Tester Channnel 1 Input.

Example See example for `AP.S2DSP.MLS.Ch1Rdg`.

AP.S2DSP.MLS.Ch1Trig ❷ Method

Syntax **`AP.S2DSP.MLS.Ch1Trig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.MLS.Ch1Rdg` command. The reading in progress is aborted.

See Also `AP.S2DSP.MLS.Ch1Rdg, AP.S2DSP.MLS.Ch1Ready`

Example See example for `AP.S2DSP.MLS.Ch1Rdg`.

AP.S2DSP.MLS.Ch2Rdg ❷ Property

Syntax **AP.S2DSP.MLS.Ch2Rdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	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 Quasi-Anechoic Acoustical Tester channel 2 Peak Monitor meter and zeros the ready count.

See Also AP.S2DSP.MLS.Ch2Ready, AP.S2DSP.MLS.Ch2Trig

Example See example for AP.S2DSP.MLS.Ch1Rdg.

AP.S2DSP.MLS.Ch2Ready ❷ Property

Syntax **AP.S2DSP.MLS.Ch2Ready**

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Quasi-Anechoic Acoustical Tester channel 2 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 AP.S2DSP.MLS.Ch2Rdg or AP.S2DSP.MLS.Ch2Trig commands will zero the ready count.

If the reading is found to be ready, a call to the AP.S2DSP.MLS.Ch2Rdg command will be guaranteed to return quickly.

See Also AP.S2DSP.MLS.Ch2Rdg, AP.S2DSP.MLS.Ch2Trig

Example See example for AP.S2DSP.MLS.Ch1Rdg.

AP.S2DSP.MLS.Ch2Source ❷ Property

Syntax `AP.S2DSP.MLS.Ch2Source`

Data Type Integer

The following list contains the selections relevant to the `AP.S2DSP.MLS.InputFormat` command A/D input selection.

0	Anlr-A
1	Anlr-B
2	Anlr Reading Ampl
3	Anlr Reading Ratio
4	Ch. A Generator
5	Ch. B Generator
6	Jitter Signal
7	None

The following list contains the selections relevant to the `AP.S2DSP.MLS.InputFormat` command Digital input selection.

0	A
1	B
2	None

Description This command sets the Quasi-Anechoic Acoustical Tester Channnel 2 Input.

Example See example for `AP.S2DSP.MLS.Ch1Rdg`.

AP.S2DSP.MLS.Ch2Trig ❷ Method

Syntax `AP.S2DSP.MLS.Ch2Trig`

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.S2DSP.MLS.Ch2Rdg` comand. The reading in progress is aborted.

See Also AP.S2DSP.MLS.Ch2Rdg, AP.S2DSP.MLS.Ch2Ready

Example See example for AP.S2DSP.MLS.Ch1Rdg.

AP.S2DSP.MLS.InputFormat ❷ Property

Syntax **AP.S2DSP.MLS.InputFormat**

Data Type Integer

0	Digital: MLS can acquire signals (two channels) directly from any of the digital interfaces of System One Dual Domain.
1	Low BW (1x) A/D: To measure analog domain signals up to 24 kHz with maximum dynamic range and frequency resolution, select Low BW (1x) A/D. The 1x indicates that these A/D converters operate directly at the Internal Sample Rate.

Description This command sets the Quasi-Anechoic Acoustical Tester Input Format.

Example See example for AP.S2DSP.MLS.Ch1Rdg.

AP.S2DSP.MLS.TimeDelay ❷ Property

Syntax **AP.S2DSP.MLS.TimeDelay(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following unit is valid for this command: sec

Description This command sets the Quasi-Anechoic Acoustical Tester Time Delay.

The Time Delay field is used to tell the DSP the distance from the speaker under test to the measurement microphone as a reference for the phase measurements. This information allows the DSP to subtract out the transit time delay from the phase readings. As the Time Delay value is adjusted the phase response will slope up or down reflecting the constant time delay component of the data. The initial value of

Time Delay may be estimated from a measurement of the distance between loudspeaker and microphone. The proper final Time Delay value may be determined experimentally as the peak amplitude on a time domain graph or to obtain the smallest slope on phase.

Example

```
Sub Main
   AP.Prompt.FontSize = 8  'Set font size to 8 point.
   AP.Prompt.Position(-1,-1,220,130) 'Set location and _
      size.
   AP.Application.NewTest
   AP.Gen.Wfm 7, 0
   AP.Gen.Output = True
   AP.Anlr.ChAInput = 2
   AP.Anlr.ChBInput = 2
   AP.Application.PanelOpen apbPanelDSPSmall
   AP.S2Dsp.Program = 5
   AP.S2Dsp.Mls.InputFormat = 1
   AP.S2Dsp.Mls.TimeDisplay = 1  'Energy-Time display
   AP.S2Dsp.Mls.WindowETime = 1  'Select Half Hann _
      Energy-Time Window
   AP.S2Dsp.Mls.TrigSource = 1   'Analog Generator
   AP.S2Dsp.Mls.WfmDisplay = 0   'Interpolate
   AP.Application.PanelOpen apbPanelSweepSmall
   AP.Sweep.Data1.Id = 6326
   AP.Sweep.Source1.Id = 5582
   AP.Sweep.Stereo = True
   AP.Sweep.Source1.Stop("sec") = 0.005
   AP.Sweep.Start                'Run Sweep
   AP.Graph.OptimizeLeft         'Optimize display _
      for Data 1

   AP.Prompt.Text = "Energy-Time Response."
   AP.Prompt.ShowWithContinue    'Display prompt with _
      Continue button.
   Stop                          'Stop macro.

   AP.Sweep.Stereo = False
   AP.Sweep.Source1.Id = 5581    'Amplitude
   AP.Sweep.Data1.Top("dBV") = 26.020600
   AP.Sweep.Source1.LogLin = 0
   AP.Sweep.Data2.Id = 6046      'Phase
```

```
   AP.Sweep.Stereo = True
   AP.S2Dsp.Mls.TimeDelay("sec") = 38.74e-6
   AP.S2Dsp.Mls.WindowStart = 0  'None
   AP.S2Dsp.Mls.WindowStop = 0   'None
   AP.Sweep.Retransform          'Retransform FFT
   AP.Graph.OptimizeIndividually 'Optimize display

   AP.Prompt.Text = "Frequency and Phase Response."
   AP.Prompt.ShowWithContinue    'Display prompt with _
      Continue button.
   Stop                          'Stop macro.
End Sub
```


AP.S2DSP.MLS.TimeDisplay ❷ Property

Syntax `AP.S2DSP.MLS.TimeDisplay`

Data Type Integer

0	Impulse Response: will show the results of the MLS correlation which is the actual impulse response of the device under test.
1	Energy-Time: will display what is commonly called an energy-time curve. The energy-time curve computation process involves transforming the impulse response to the frequency domain, doing further processing in the frequency domain, and transforming the result back to the time domain. A frequency window may be used for the conversion from frequency domain back to time domain. The frequency window is selected in the Energy-Time Window field.

Description This command sets the Quasi-Anechoic Acoustical Tester Time Domain Display type.

Example See example for `AP.S2DSP.Mls.TimeDelay`.

AP.S2DSP.MLS.TrigSource ❷ Property

Syntax `AP.S2DSP.MLS.TrigSource`

Data Type Integer

0	Analog Generator
1	Digital Generator

Description This command sets the Quasi-Anechoic Acoustical Tester Trigger Source.

Example See example for `AP.S2DSP.Mls.TimeDelay`.

AP.S2DSP.MLS.WfmDisplay ❷ Property

Syntax `AP.S2DSP.MLS.WfmDisplay`

Data Type Integer

0	Interpolate
1	Display Samples
2	Peak Values

Description This command sets the Quasi-Anechoic Acoustical Tester waveform display mode.

When Interpolate is selected, the DSP will compute the data value, interpolated from the nearby measured values. This smooths out the stair-step appearance of frequency response curves at low frequencies with a Log horizontal axis, where the bin width (usually 2.93 Hz at the 48 kHz sample rate) occupies a significant portion of the screen.

When Display Samples is selected, the DSP will return the closest actual measured value without altering the data. Normal is the recommended display mode for frequency response data with a Linear horizontal axis or with a Log axis above 100 to 300 Hz. In these cases, the jagged lines caused by the FFT bin width are not usually noticeable.

When Peak Values is selected, The Peak mode will return the largest value between the last requested sweep point and the current one. Peak is recommended for time domain MLS displays (Impulse

Response and Energy-Time). Peak mode would not normally be used for frequency response displays with MLS.AZ1, since high values are of no more interest than low values when plotting frequency response.

Example See example for `AP.S2DSP.Mls.TimeDelay`.

AP.S2DSP.MLS.WindowETime ❷ Property

Syntax `AP.S2DSP.MLS.WindowETime`

Data Type Integer

0	No Window: will perform the required transformations with all frequency components of the signal included in the computations.
1	Half Hann: reduces the contribution of high frequencies. The low frequency information remains unchanged. When operating at the 48 kHz sample rate this window filters out energy above 12 kHz.
2	Hann: reduces both high and low frequency energy, concentrating on arrivals at the center of the frequency range. Since the processing occurs on a linear frequency scale, this will focus analysis on signals around one quarter of the sample rate. At 48 kHz this will result in the 12 kHz energy dominating the energy-time display. This selection is not fundamentally useful for most applications, but is included for correlation to measurements by other manufacturers
3	<240Hz >8kHz: filters out energy below 240 Hz and above 8 kHz, producing equal sensitivity to signals over a 5 octave range.
4	<124Hz >16kHz: spreads the analysis over a 7 octave range.

Description This command sets the Quasi-Anechoic Acoustical Tester Energy-Time Window selection.

Example See example for `AP.S2DSP.Mls.TimeDelay`.

AP.S2DSP.MLS.WindowStart ❷ Property

Syntax `AP.S2DSP.MLS.WindowStart`

Data Type Integer

0	None:
1	<5%
2	<10%
3	<20%
4	<30%

Description This command sets the Quasi-Anechoic Acoustical Tester Start Time Window selection.

When a section of the impulse response (direct arrival signal before reflections, for example) is isolated and transformed into the frequency domain, the impulse amplitude at the beginning and ending of that section will generally not be exactly the same and thus will not splice smoothly. The sharp edges introduced into the impulse response by splicing unequal amplitudes will produce ripples in the resulting frequency response plot. Windowing the time domain data by attenuating the amplitude at the beginning and end of the section to be transformed will reduce this rippling, but also reduces the steepness of transitions in the frequency response plots. The Time Start Window and Time Stop Window fields select the window applied to the impulse response (time domain) when transforming it to the frequency domain.

The time window is made up of two half-windows. The first half is selected in the Time Start Window field and is used to process the first portion of data, beginning at the Source 1 Start time on the Sweep panel. The second half-window is selected in the Time Stop Window field and processes the later portion of data, ending at the selected Stop time on the Sweep panel. Separate selection of the Source 1 Start and Stop half-windows permits creation of asymmetrical windows, which provide the optimum match to the asymmetrical shape of the typical impulse response. To change selections, click on the down arrow at the right of the field and click on the desired selection in the list which is displayed. The available selections at both the Time Start Window and Time Stop Window fields are a family of half-cycle raised cosine functions labeled NONE, <5%, <10%, <20%

and <30%. The numeric value refers to the amount of the data record (time span multiplied by sample period) taken up by the window's transition from zero to full amplitude. The Time Start Window half-window starts with an amplitude of zero at the Sweep panel Start time and climbs to an amplitude of 1.00 (no attenuation) at or before the selected percentage of the record. The Time Stop Window half-window starts with an amplitude of 1.00 at or following a point during the record which is within the selected percentage of the record end, and falls to zero at the Sweep panel Stop time. The windows with a steeper transition will alter the data less but will also have less impact on the frequency response ripples. The more gradual transitions have greater ripple reduction but alter the data more.

Example See example for `AP.S2DSP.Mls.TimeDelay`.

AP.S2DSP.MLS.WindowStop ❷ Property

Syntax `AP.S2DSP.MLS.WindowStop`

Data Type Integer

0	None:
1	<5%
2	<10%
3	<20%
4	<30%

Description This command sets the Quasi-Anechoic Acoustical Tester Stop Time Window selection.

See Also `AP.S2DSP.MLS.WindowStart`

Example See example for `AP.S2DSP.Mls.TimeDelay`.

User Notes

User Notes

User Notes

User Notes

User Notes

System Two DSP Program & Reference

AP.S2DSP.Program ❷ Property

Syntax `AP.S2DSP.Program`

Data Type Integer

0	None: No Digital Analyzer selected.
1	DSP Audio Analyzer (ANALYZER): usable only for digital domain input signals, with the SYS-2300 series of models. Measures frequency, amplitude (on both stereo channels simultaneously), the ratio of amplitudes on the two stereo channels, selective amplitude, crosstalk between channels, THD+N with either ratio units (% and dB) or absolute units, and noise unweighted, A-weighted, or CCIR-468 weighted with RMS or quasi-peak detectors. ANALYZER is the approximate System Two equivalent to the analysis functions of the GENANLR.DSP program for System One.
2	FFT spectrum analyzer (FFT): usable for analog domain input signals with SYS-2200 or SYS-2300 models, and for digital domain input signals with SYS-2300 models. Provides general-purpose time domain (oscilloscope) display of waveforms or frequency domain (spectrum analyzer) display of signals, including the received jitter signal on Dual Domain units. Features include double precision transforms for better than 140 dB dynamic range, pre-trigger, a variety of selectable transform lengths, the ability to position the start of the transformed section anywhere in the acquired record, FFT power-law averaging, four windowing functions, and several types of waveform processing for display. FFT for System Two combines the functions of FFTSLIDE.DSP and the analysis functions of FFTGEN.DSP for System One, with greatly improved dynamic range due to double precision FFT computations, an additional windowing function not available

with System One, and power-law averaging for improved accuracy on noise signals.

3 Digital interface analyzer (INTERVU): analyzes the AES/EBU or consumer digital interface input signal of SYS-2300 series models via a 67 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.

4 Multitone audio analyzer (FASTTEST): usable for analog (SYS-2200 or SYS-2300 models) or digital domain (SYS-2300 models) 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, 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 imbedded psychoacoustic model of the frequency masking effect in humans. FASTTEST for System Two combines the features of FASTTEST.DSP, FASTTRIG.DSP, and CODEC.DSP for System One.

5 Quasi-Anechoic Acoustical Tester (MLS): The Quasi-Anechoic Acoustical Tester (MLS) program for the

Digital Analyzer uses Maximum Length Sequence (MLS) testing to characterize the linear response of acoustical and electronic devices. It permits time-selective measurements in which one signal, such as the direct sound from a loudspeaker, may be separated from another similar signal, such as a room reflection. The time window may be adjusted to allow measurement of any arrival in a complex reverberation pattern. These signals may be examined in the time domain (showing energy as a function of time) or in the frequency domain (amplitude and phase vs frequency). Impulse responses may be saved to disk for later down-load to the DSP and further analysis.

Except in repetitive testing with unchanged dimensions between loudspeaker under test, measurement microphone, and reflecting surfaces, use of MLS typically involves both time domain and frequency domain displays. It is normally necessary to examine the time domain impulse response from MLS to determine the exact arrival time of the signal and the first reflection, designation of that time section for FFT spectrum analysis, and finally graphing of the anechoic frequency (and possibly phase) response for examination or comparison to limits.

Description This command selects a System Two Digital Analyzer type.

See Also `AP.S1DSP.Program`

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.DGen.Wfm(2,0)  'Put DGen in IMD Mode (SMPTE1:1).
   AP.DGen.Freq ("Hz")  = 2000 'Set sine wave frequency.
   AP.DGen.IMFreq ("Hz") = 80 'Set IM freq to 80 Hz.
   AP.S2DSP.Program = 1 'Digital Domain Audio Analyzer.
   AP.S2DSP.Analyzer.FuncSettling 1.0, 1e-3, "V", 3, _
      30e-3, 1
   AP.DGen.Output = True      'Turn on output.
   For ratio = 51.0 To 1.00 Step -10   'Increment ratio.
      AP.DGen.AmplRatio("%") = ratio
      AP.S2DSP.Analyzer.FuncTrig
      While AP.S2DSP.Analyzer.FuncReady = 0
```

```
      Wend
      msg = msg & "Reading("&ratio&"%) = " & _
         AP.S2DSP.Analyzer.FuncRdg ("V") & Chr(13)
   Next
   AP.Prompt.Text = msg
   AP.Prompt.ShowWithContinue
   Stop
End Sub
```

AP.S2DSP.RefCh1dBr ❷ Property

Syntax **AP.S2DSP.RefCh1dBr(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dbFS, Bits, V, Vp, Vpp, dBu, dBV, Hz

Description This command sets the dBr1 value as the reference for the dBr1 unit selectable at the Level Monitor meter and at the main Function meter in absolute functions. When analog domain units are selected for the dBr1 unit, is are converted into the digital domain via the V/FS Reference value.

Example

```
Sub Main
   AP.File.OpenTest "Ref1.at2"
   AP.S2DSP.RefVFS("V") = 2
   AP.S2DSP.RefFreq("Hz") = 2000
   AP.S2DSP.RefCh1dBr("FFS") = .5
   AP.S2DSP.RefCh2dBr("FFS") = .75
   Wait .5
   'Get new Ch A Freq reading
   F_reading1 = AP.S2DSP.Analyzer.ChAFreqRdg("%Hz")
   'Get new Ch A Level reading
   L_reading1 = AP.S2DSP.Analyzer.ChALevelRdg("dBr2")
   A_reading1 = AP.S2DSP.Analyzer.FuncRdg("dBr1")
   V_reading1 = AP.S2DSP.Analyzer.FuncRdg("V")
   NewLine$ = Chr(13)
```

```
    a$= “Ch A Level Reading _
       “+Left(Str$(L_reading1),6)+”dBr 2"
    b$= “Ch A Freq Reading ”+Left(Str$(F_reading1),6) _
       +"%Hz"
    c$= “Function Meter Reading _
       “+Left(Str$(A_reading1),6)+”dBr 1"
    d$= “Function Meter Reading _
       “+Left(Str$(V_reading1),6)+”V/FS"
    AP.Prompt.Text = a$ + NewLine$ + b$ + NewLine$ + c$ _
       + NewLine$ + d$
    AP.Prompt.ShowWithContinue
    Beep
    Stop
End Sub
```

AP.S2DSP.RefCh2dBr

❷ Property

Syntax **AP.S2DSP.RefCh2dBr(*unit$*)**

Data Type Double

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: FFS, %FS, dbFS, Bits, V, Vp, Vpp, dBu, dBV, Hz

Description This command sets the dBr2 value as the reference for the dBr2 unit selectable at the Level Monitor meter and at the main Function meter in absolute functions. When analog domain units are selected for the dBr2 unit, is are converted into the digital domain via the V/FS Reference value.

Example See example for `AP.S2DSP.RefCh1dBr`.

AP.S2DSP.RefFreq

❷ Property

Syntax **AP.S2DSP.RefFreq(*unit$*)**

Data Type Double

Parameters	Name	Description
	unit$	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 Digital Analyzer Frequency counter.

Example See example for `AP.S2DSP.RefCh1dBr`.

AP.S2DSP.RefVFS ❷ Property

Syntax **`AP.S2DSP.RefVFS(`*`unit$`*`)`**

Data Type Double

Parameters	Name	Description
	unit$	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, dBr1, or dBr2 to express the measured digital amplitude in terms of the analog input value to the converter.

Example See example for `AP.S2DSP.RefCh1dBr`.

User Notes

User Notes

User Notes

User Notes

Speaker

AP.Speaker.Mode ❷ Property

Syntax `AP.Speaker.Mode`

Data Type Integer

0	Mono: a single signal is fed to both left and right headphones and to the internal loudspeaker.
1	Stereo: different signals are fed to the left and right headphones (in most cases). Both these signals are summed into the internal monaural loudspeaker located in the bottom of the instrument.

Description This command selects the output configuration for the speaker output jack.

See Also `AP.Speaker.Source`

Example

```
Sub Main
   AP.App.NewTest
Start:
   Begin Dialog UserDialog 280,133
      PushButton 20,14,240,28,"Monitor Analog Generetor _
         Channel A",.PushButton1
      PushButton 20,49,240,28,"Monitor Analog Generetor _
         Channel B",.PushButton2
      PushButton 20,91,240,28,"EXIT (Speaker _
         OFF)",.PushButton3
   End Dialog
   Dim dlg As UserDialog

   Dim MainMenu As UserDialog
   Select Case Dialog(MainMenu)
      Case 1
         AP.Speaker.Mode = 0      'Mono
         AP.Speaker.Source = 1    'Analog Generator Ch A
      Case 2
```

```
            AP.Speaker.Mode = 0
            AP.Speaker.Source = 2    'Analog Generator Ch B
         Case Else
            AP.Speaker.Mode = 0
            AP.Speaker.Source = 0    'Speaker OFF
            End
   End Select
   GoTo Start:
End Sub
```

AP.Speaker.Source

❷ Property

Syntax `AP.Speaker.Source`

Data Type Integer

The following list contains the selections relevant to the `AP.Speaker.Mode` command Mono Configuration.

0	Off: disables audible monitoring.
1	Reading: is the final analog signal in the Analog Analyzer, following all filtering (and following the wow and flutter discriminator or IMD detectors if the reading meter is in W&F or IMD modes).
2	Generator Monitor A:
3	Generator Monitor B:
4	DSP Monitor A:
5	DSP Monitor B:
6	Analog Input A:
7	Analog Input B:

The following list contains the selections relevant to the `AP.Speaker.Mode` command Stereo Configuration.

0	Off: disables audible monitoring.
1	Reading: is the final analog signal in the Analog Analyzer, following all filtering (and following the wow and flutter discriminator or IMD detectors if the reading meter is in W&F or IMD modes).

2	Generator Monitor:
3	Analog Input A:

Description This command selects a monitoring location(s) for the speaker and headphone jack outputs.

See Also `AP.Speaker.Mode`

Example See example for `AP.Speaker.Mode`.

User Notes

User Notes

User Notes

Sweep

AP.Sweep.Append ❶❷ Property

Syntax **AP.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 AP.Sweep.Repeat, AP.Sweep.StartWithAppend, AP.Sweep.StartWithRepeat

Example

```
Sub Main
   AP.Application.NewTest  'Reset panels
   AP.Sweep.CreateGraph = 1
   AP.Sweep.CreateTable = 0
   AP.Sweep.GraphType = 0
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 2
   AP.Anlr.FuncFilterHP = 3
   AP.Anlr.FuncFilterLP = 0

   'The commands in the following section could be
   '  replaced with commands for Data1-6
   AP.Sweep.Data1.Id = 5906
   AP.Sweep.Data1.Limits("None",1,1)

   'The commands in the following section could be
   '  replaced with commands for Data2
   AP.Sweep.Data1.AutoDiv = 0
   AP.Sweep.Data1.Div = 1
   AP.Sweep.Data1.Autoscale = 1
```

```
   AP.Sweep.Data1.LogLin = 1
   AP.Sweep.Data1.Top("V") = 1
   AP.Sweep.Data1.Bottom("V") = 0

   AP.Sweep.Source1.Start("Hz") = 20.0
   AP.Sweep.Source1.Stop("Hz") = 200000.0
   AP.Sweep.PreSweepDelay = 0.2

   AP.Sweep.Start
   AP.Sweep.Append = True

   AP.Anlr.FuncFilterHP = 2
   AP.Anlr.FuncFilterLP = 1
   AP.Sweep.Start
   AP.Anlr.FuncFilterHP = 1
   AP.Anlr.FuncFilterLP = 2
   AP.Sweep.Start
   AP.Anlr.FuncFilterHP = 0
   AP.Anlr.FuncFilterLP = 3

   AP.Data.OptimizeDisplay(0)
End Sub
```


AP.Sweep.CopyData1To2 ❶❷ Method

Syntax **AP.Sweep.CopyData1To2**

Data Type Boolean

Description This command copies the Sweep panel Data 1 settings to Data 2

❷ Example

```
Sub Main
   AP.Application.NewTest  'New Test
   AP.Gen.Output = True 'Generator Output ON
   AP.Anlr.ChAInput = 2 'Ch A Input to GenMon
   AP.Anlr.ChBInput = 2 'Ch B Input to GenMon
   AP.Anlr.FuncMode = 3 'Func Meter to THD+N Ampl

   AP.S2Dsp.Program = 2 'Select FFT Digital Analyzer
   AP.S2Dsp.FFT.InputFormat = 1'Select Low BW A/D Input
```

```
    AP.S2Dsp.FFT.Ch1Source = 2 'Digital Analyzer Ch 1 _
       Source to Anlr Rdg Ampl

    AP.Sweep.Data1.Id = 6024   'Select Fft.Ch.1 Ampl _
       for Data 1
    AP.Sweep.Data2.Id = 6027   'Select Fft.Ch.2 Ampl _
       for Data 2
    AP.Sweep.Source1.Id = 5515 'Select Fft.FFT Freq. _
       for Source 1
    AP.Sweep.Start             'Acqure waveform
    'Display data so that the vertical scaleing is _
       relative to optimized data for Data 1
    AP.Graph.OptimizeLeft      'Optimize Data 1
    AP.Graph.CopyToSweepPanel  'Copy Left and Right _
       graph vertical scale information to Sweep Panel
    AP.Sweep.CopyData1to2      'Copy Data 1 settings _
       to Data 2
    Wait 5
    'Display data so that the vertical scaleing is _
       relative to optimized data for Data 2
    AP.Graph.OptimizeRight     'Optimize Data 2
    AP.Graph.CopyToSweepPanel  'Copy Left and Right _
       graph vertical scale information to Sweep Panel
    AP.Sweep.CopyData2to1      'Copy Data 2 settings
       to Data 1
End Sub
```


AP.Sweep.CopyData2To1

12 Method

Syntax `AP.Sweep.CopyData2To1`

Data Type Boolean

Description This command copies the Sweep panel Data 2 settings to Data 1

Example See example for `AP.Sweep.CopyData1To2`.

AP.Sweep.CreateGraph

12 Property

Syntax `AP.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 `AP.Sweep.CreateTable, AP.Sweep.GraphType`

Example See example for `AP.Sweep.Append`.

AP.Sweep.CreateTable

12 Property

Syntax `AP.Sweep.CreateTable`

Data Type Boolean

True	Display a Data Table window when starting the sweep if a Data Table window is not displayed.
False	Do Not create a Data Table window when starting the sweep.

Description This command enables or disables creation of the Data Table window when a sweep is run.

See Also `AP.Sweep.CreateGraph, AP.Sweep.GraphType`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data1.AutoDiv ❶❷ Property

Syntax `AP.Sweep.Data1.AutoDiv`

Data Type Boolean

True	Automatically select the number of divisions.
False	Use the number of divisions defined by the AP.Sweep.Data1.Div command.

Description This command enables or disables automatic selection of the number of linear vertical axis divisions displayed for Data 1.

See Also `AP.Sweep.Data1.Div, AP.Sweep.Data1.LogLin`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data1.Autoscale ❶❷ Property

Syntax `AP.Sweep.Data1.Autoscale`

Data Type Boolean

True	Autoscale graph vertical axis for Data 1.
False	Do Not Autoscale graph vertical axis for Data 1.

Description This command enables or disables automatic scaling of the graph vertical axis Top and Bottom values for Data 1. The Data 1 vertical axis is shown on the left side of the graph.

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data1.Bottom ❶❷ Property

Syntax `AP.Sweep.Data1.Bottom(unit$)`

Data Type Double — Enter a value that is to be displayed at the bottom of the graph left axis.

Parameters

Name	Description

	unit$	Refer to the setting or reading defined by the `AP.Sweep.Data1.Id` command to determine the appropriate unit selections.

Description This command defines the bottom value on the graph vertical axis located on the left side of the graph window.

See Also `AP.Sweep.Data1.Top`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data1.Div

01 02 Property

Syntax **`AP.Sweep.Data1.Div`**

Data Type Long — Number of divisions displayed.

Description This command sets the number of divisions that are to be displayed for a linear vertical axis defined on Data 1. The `AP.Sweep.Data1.AutoDiv` must be disabled.

See Also `AP.Sweep.Data1.AutoDiv, AP.Sweep.Data1.LogLin`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data1.Id

01 02 Property

Syntax **`AP.Sweep.Data1.Id`**

Data Type Long — Instrument Parameter ID#.

Description This command is used to select the instrument parameter, which will return readings for Data 1.

Refer to Appendix D to obtain instrument parameter identification numbers.

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data1.Limits 01 02 Method

Syntax `AP.Sweep.Data1.Limits(filename$, column%, upper)`

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN limit file (.adl). Enter "None" for the file name to remove the limit file from Data 1.
column%	1 = Data 1 measurements. 2 = Data 2 measurements. 3 = Data 3 measurements. 4 = Data 4 measurements. 5 = Data 5 measurements. 6 = Data 6 measurements.
upper	True = Upper Limit. False = Lower Limit.

Result Boolean

True	File attachment successfull.
False	File attachment failed.

Description This command attaches or removes a limit file from Data 1 for upper or lower limit comparisions.

See Also `AP.Sweep.Recompare`

Example

```
Sub Main
   AP.Log.Enable = 0    'Disable log file
   AP.File.OpenTest "CODEC.AT1" 'Open test to create a _
      masking curve limit file.
   AP.Sweep.Start             'Start sweep.
   AP.File.SaveDataAs "MASK.ADL" 'Save masking curve as _
      MASK.ADL limit file.
   AP.Application.NewData     'Remove Masking curve _
      data from memory.
   AP.S1DSP.Codec.Mode = 0    'Set the DSP panel _
      measurement field to spectrum mode.
   AP.Sweep.Reprocess         'Reprocess the acquired _
      waveform and display spectrum results.
   AP.Sweep.Source1.Table ("CODEC.ADS",0)
                              'Attach Sweep table.
```

```
   AP.S1DSP.Codec.Mode = 1 'Set the DSP panel _
      measurement field to response mode.
   AP.Sweep.Reprocess      'Reprocess the acquired _
      waveform and display response results.

'Attach upper limits to Data1 & Data3.
   AP.Sweep.Data1.Limits("MASK.ADL", 1, True)
   AP.Sweep.Data3.Limits("Mask.ADL", 1, True)

   AP.S1DSP.Codec.Mode = 2 'Set DSP measurement mode _
      to distortion.
   AP.Sweep.Reprocess         'Reprocess the acquired _
      waveform and display distortion results.
   AP.S1DSP.Codec.Mode = 3    'Set the DSP panel _
      measurement field to noise mode.
   AP.Sweep.Reprocess         'Reprocess the acquired _
      waveform and display noise results.
End Sub
```


AP.Sweep.Data1.LogLin

❶❷ Property

Syntax `AP.Sweep.Data1.LogLin`

Data Type Integer

0	Logarithmic vertical axis.
1	Linear vertical axis.

Description This command determines the Data 1 vertical axis data scaling type.

See Also `AP.Sweep.Data1.Div, AP.Sweep.Data1.AutoDiv`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data1.Top

❶❷ Property

Syntax `AP.Sweep.Data1.Top(`*unit$*`)`

Data Type Double

Enter a value that is to be displayed at the top of the graph left axis.

Parameters	Name	Description
	unit$	Refer to the setting or reading defined by the AP.Sweep.Data1.Id command to determine the appropriate unit selections.

Description This command defines the top value on the graph vertical axis located on the left side of the graph window.

See Also AP.Sweep.Data1.Bottom

Example See example for AP.Sweep.Append.

AP.Sweep.Data2.AutoDiv

❶❷ Property

Syntax **AP.Sweep.Data2.AutoDiv**

Data Type Boolean

True	Automatically select the number of divisions.
False	Use the number of divisions defined by the AP.Sweep.Data2.Div command.

Description This command enables or disables automatic selection of the number of linear vertical axis divisions displayed for Data 2.

See Also AP.Sweep.Data2.Div, AP.Sweep.Data2.LogLin

Example See example for AP.Sweep.Append.

AP.Sweep.Data2.Autoscale

❶❷ Property

Syntax **AP.Sweep.Data2.Autoscale**

Data Type Boolean

True	Autoscale graph vertical axis for Data 2.

	False	Do Not Autoscale graph vertical axis for Data 2.

Description This command enables or disables automatic scaling of the graph vertical axis Top and Bottom values for Data 2. The Data 2 vertical axis is shown on the right side of the graph.

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data2.Bottom

❶❷ Property

Syntax **`AP.Sweep.Data2.Bottom(`*`unit$`*`)`**

Data Type Double — Enter a value that is to be displayed at the bottom of the graph right axis.

Parameters

Name	Description
unit$	Refer to the setting or reading defined by the `AP.Sweep.Data2.Id` command to determine the appropriate unit selections.

Description This command defines the bottom value on the graph vertical axis located on the right side of the graph window.

See Also `AP.Sweep.Data2.Top`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data2.Div

❶❷ Property

Syntax **`AP.Sweep.Data2.Div`**

Data Type Long — Number of divisions displayed.

Description This command sets the number of divisions that are to be displayed for a linear vertical axis defined on Data 2. The `AP.Sweep.Data2.AutoDiv` must be disabled.

See Also `AP.Sweep.Data2.AutoDiv, AP.Sweep.Data2.LogLin`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data2.Id

❶❷ Property

Syntax `AP.Sweep.Data2.Id`

Data Type Long — Instrument Parameter ID#.

Description This command is used to select the instrument parameter, which will return readings for Data 2.

Refer to Appendix D to obtain instrument parameter identification numbers.

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data2.Limits

❶❷ Method

Syntax `AP.Sweep.Data2.Limits(`*`filename$, column%, upper`*`)`

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN limit file (.adl). Enter "None" for the file name to remove the limit file from Data 2.
column%	1 = Data 1 measurements. 2 = Data 2 measurements. 3 = Data 3 measurements. 4 = Data 4 measurements. 5 = Data 5 measurements. 6 = Data 6 measurements.
upper	True = Upper Limit. False = Lower Limit.

Result Boolean

True	File attachment successfull.
False	File attachment failed.

Description This command attaches or removes a limit file from Data 2 for upper or lower limit comparisions.

See Also `AP.Sweep.Recompare`

Example See example for `AP.Sweep.Data1.Limits`.

AP.Sweep.Data2.LogLin

Property

Syntax **`AP.Sweep.Data2.LogLin`**

Data Type Integer

0	Logarithmic vertical axis.
1	Linear vertical axis.

Description This command determines the Data 2 vertical axis data scaling type.

See Also `AP.Sweep.Data2.Div, AP.Sweep.Data2.AutoDiv`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data2.Top

Property

Syntax **`AP.Sweep.Data2.Top(`*`unit$`*`)`**

Data Type Double — Enter a value that is to be displayed at the top of the graph right axis.

Parameters

Name	Description
unit$	Refer to the setting or reading defined by the `AP.Sweep.Data2.Id` command to determine the appropriate unit selections.

Description This command defines the top value on the graph vertical axis located on the right side of the graph window.

See Also `AP.Sweep.Data2.Bottom`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data3.Id ❶❷ Property

Syntax `AP.Sweep.Data3.Id`

Data Type Long Instrument Parameter ID#.

Description This command is used to select the instrument parameter, which will return readings for Data 3.

Refer to Appendix D to obtain instrument parameter identification numbers.

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data3.Limits ❶❷ Method

Syntax `AP.Sweep.Data3.Limits(filename$, column%, upper)`

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN limit file (.adl). Enter "None" for the file name to remove the limit file from Data 3.
column%	1 = Data 1 measurements. 2 = Data 2 measurements. 3 = Data 3 measurements. 4 = Data 4 measurements. 5 = Data 5 measurements. 6 = Data 6 measurements.
upper	True = Upper Limit. False = Lower Limit.

Result Boolean

True	File attachment successfull.
False	File attachment failed.

Description This command attaches or removes a limit file from Data 3 for upper or lower limit comparisions.

See Also `AP.Sweep.Recompare`

Example See example for `AP.Sweep.Data1.Limits`.

AP.Sweep.Data4.Id

❶❷ Property

Syntax **`AP.Sweep.Data4.Id`**

Data Type Long Instrument Parameter ID#.

Description This command is used to select the instrument parameter, which will return readings for Data 4.

Refer to Appendix D to obtain instrument parameter identification numbers.

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data4.Limits

❶❷ Method

Syntax **`AP.Sweep.Data4.Limits(`***`filename$, column%, upper`***`)`**

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN limit file (.adl). Enter "None" for the file name to remove the limit file from Data 4.
column%	1 = Data 1 measurements. 2 = Data 2 measurements. 3 = Data 3 measurements. 4 = Data 4 measurements. 5 = Data 5 measurements. 6 = Data 6 measurements.
upper	True = Upper Limit. False = Lower Limit.

Result Boolean

True	File attachment successfull.
False	File attachment failed.

Description This command attaches or removes a limit file from Data 4 for upper or lower limit comparisions.

See Also `AP.Sweep.Recompare`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data5.Id

❶❷ Property

Syntax **`AP.Sweep.Data5.Id`**

Data Type Long Instrument Parameter ID#.

Description This command is used to select the instrument parameter, which will return readings for Data 5.

Refer to Appendix D to obtain instrument parameter identification numbers.

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data5.Limits

❶❷ Method

Syntax **`AP.Sweep.Data5.Limits(`***`filename$, column%, upper`***`)`**

Parameters

Name	Description
`filename$`	Any valid DOS path and file name. The file must be an APWIN limit file (.adl). Enter “None” for the file name to remove the limit file from Data 5.
`column%`	1 = Data 1 measurements. 2 = Data 2 measurements. 3 = Data 3 measurements. 4 = Data 4 measurements. 5 = Data 5 measurements. 6 = Data 6 measurements.

	upper	True = Upper Limit. False = Lower Limit.

Result Boolean

True	File attachment successfull.
False	File attachment failed.

Description This command attaches or removes a limit file from Data 5 for upper or lower limit comparisions.

See Also `AP.Sweep.Recompare`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data6.Id

❶❷ Property

Syntax **`AP.Sweep.Data6.Id`**

Data Type Long — Instrument Parameter ID#.

Description This command is used to select the instrument parameter, which will return readings for Data 6.

Refer to Appendix D to obtain instrument parameter identification numbers.

Example See example for `AP.Sweep.Append`.

AP.Sweep.Data6.Limits

❶❷ Method

Syntax **`AP.Sweep.Data6.Limits(`***`filename$, column%, upper`***`)`**

Data Type Boolean

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN limit file (.adl). Enter "None" for the file name to remove the limit file from Data 6.
column%	1 = Data 1 measurements. 2 = Data 2 measurements.

3 = Data 3 measurements.
4 = Data 4 measurements.
5 = Data 5 measurements.
6 = Data 6 measurements.

upper	True = Upper Limit. False = Lower Limit.

Result Boolean

True	File attachment successfull.
False	File attachment failed.

Description This command attaches or removes a limit file from Data 6 for upper or lower limit comparisions.

See Also `AP.Sweep.Recompare`

Example See example for `AP.Sweep.Append`.

AP.Sweep.GraphType

❶❷ Property

Syntax **`AP.Sweep.GraphType`**

Data Type Integer

0	X - Y mode. Data 1-6 measurements are displayed on the vertical axis and Source settings are displayed on the horizontal axis.
1	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 `AP.Sweep.Data2.Id` must be defined.

See Also `AP.Sweep.Data2.Id`

Example See example for `AP.Sweep.Append`.

AP.Sweep.PreSweepDelay

❶❷ Property

Syntax `AP.Sweep.PreSweepDelay`

Data Type Double 0.0 to 3.0 sec.

Parameters None

Description This command sets a user-controllable time delay value inserted after the `AP.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 `AP.Sweep.Start`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Recompare

❶❷ Method

Syntax `AP.Sweep.Recompare`

Parameters None

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 `AP.Sweep.Data(`*n*`).Limits` command.

This command is equivalent to F7 in APWIN.

See Also AP.Sweep.Data1.Limits, AP.Sweep.Data2.Limits, AP.Sweep.Data3.Limits, AP.Sweep.Data4.Limits, AP.Sweep.Data5.Limits, AP.Sweep.Data6.Limits

Example

AP.Sweep.Repeat ❶❷ Property

Syntax **AP.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 AP.Sweep.Append, AP.Sweep.StartWithAppend, AP.Sweep.StartWithRepeat

Example

```
Sub Main
   AP.Application.NewTest     'Start with New Test
   AP.Gen.Output = True       'Generator Output ON
   AP.Anlr.ChAInput = 2
   AP.Application.PanelOpen apbPanelSweepSmall _
      'Display Sweep Panel

   AP.Prompt.Text = "Press Continue to Stop _
      Sweep."  'Prompt text
   AP.Prompt.FontSize = 8     'Set font size to 8 point
   AP.Prompt.Position(-1,-1,190,120)'Location and size

   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
         End
         AP.Sweep.Append = False
         AP.Sweep.Repeat = False
      Case 1   'Run single sweep
         AP.Sweep.Append = False
         AP.Sweep.Repeat = False
         AP.Sweep.Start
      Case 2   'Run sweep and append data
         AP.Sweep.Repeat = False
         AP.Sweep.StartWithAppend
      Case 3   'Run repeating sweep
         AP.Sweep.Append False
   'Display prompt
         AP.Prompt.ShowWithContinueAndStopSweep
         AP.Sweep.StartWithRepeat          'Start sweep
         AP.Sweep.Repeat = False
      Case 4   'Run repeating sweep and append data
         AP.Sweep.Append = True
         AP.Sweep.Repeat = True
   'Display prompt
         AP.Prompt.ShowWithContinueAndStopSweep
         AP.Sweep.Start                    'Start sweep
         AP.Sweep.Repeat = False
   End Select
   GoTo DisplayDialog
End Sub
```

AP.Sweep.Reprocess ❶❷ Method

Syntax `AP.Sweep.Reprocess`

Parameters None

Result Boolean

True	Reprocess successful.
False	Reprocess failed.

Description This command instructs APWIN 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 APWIN software.

This command is equivalent to Ctrl+F6 in APWIN.

❶ Example

```
Sub Main
   AP.Log.Enable = 0    'Disable log file
   AP.File.OpenTest "CODEC.AT1" 'Open test to create a _
      masking curve limit file.
   AP.Sweep.Start             'Start sweep.
   AP.File.SaveDataAs "MASK.ADL" 'Save masking curve as _
      MASK.ADL limit file.
   AP.Application.NewData  'Remove Masking curve data _
       from memory.
   AP.S1DSP.Codec.Mode = 0 'Set the DSP panel _
      measurement field to spectrum mode.
   AP.Sweep.Reprocess         'Reprocess the acquired _
      waveform and display spectrum results.
                              'Attach Sweep table.
   AP.Sweep.Source1.Table ("CODEC.ADS",0)
```

```
   AP.S1DSP.Codec.Mode = 1    'Set the DSP panel _
      measurement field to response mode.
   AP.Sweep.Reprocess         'Reprocess the acquired _
      waveform and display response results.

'Attach upper limits to Data1 & Data3.
   AP.Sweep.Data1.Limits("MASK.ADL", 1, True)
   AP.Sweep.Data3.Limits("Mask.ADL", 1, True)

   AP.S1DSP.Codec.Mode = 2    'Set DSP measurement _
      mode to distortion.
   AP.Sweep.Reprocess         'Reprocess the acquired _
      waveform and display distortion results.
   AP.S1DSP.Codec.Mode = 3    'Set the DSP panel _
      measurement field to noise mode.
   AP.Sweep.Reprocess         'Reprocess the acquired _
      waveform and display noise results.
End Sub
```


AP.Sweep.Retransform

❶❷ Method

Syntax `AP.Sweep.Retransform`

Parameters None

Result Boolean

True	Retransform successful.
False	Retransform failed.

Description This command instructs APWIN 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 APWIN software.

This command is equivalent to F6 in APWIN.

❷ **Example**

```
Sub Main
   AP.App.NewTest
   AP.Gen.Output = 1
   AP.Anlr.ChAInput = 2
   AP.S2DSP.Program = 2       'Select FFT program
   AP.S2DSP.FFT.InputFormat = 1'Select Low BW(A/D) input
   AP.S2DSP.FFT.Length = 6    'Set FFT length to 16384
   AP.S2DSP.FFT.Window = 0    'Set FFT window to BH4
   AP.Sweep.Data1.Id = 6023   'Set sweep panel Data 1 _
      to Fft.Ch.1 Ampl
   AP.Sweep.Source1.Id = 5515 'Set sweep panel _
      Source 1 to Fft.FFT Freq
   AP.Sweep.Start
   AP.Sweep.Append = 1
   AP.S2DSP.FFT.Window = 1 'Set FFT window to Hann
   AP.Sweep.Retransform
   AP.S2DSP.FFT.Window = 2 'Set FFT window to Flat-Top
   AP.Sweep.Retransform
   AP.S2DSP.FFT.Window = 3 'Set FFT window to Equiripple
   AP.Sweep.Retransform
   AP.S2DSP.FFT.Window = 4 'Set FFT window to None
   AP.Sweep.Retransform
   AP.Data.OptimizeDisplay 0
End Sub
```

AP.Sweep.ReverseChannels ❶❷ Method

Syntax **AP.Sweep.ReverseChannels(***reversed***)**

Parameters

Name	Description
reversed	True = Change channel to alternate channel. False = Return channel to previous state.

Result None

Description This command selects the alternate channel from the present settings for the generator output and analyzer Function meter input selection. If channel A is selected for the generator output and the analyzer Function meter and this command is executed using a 1 for the command argument channel B will be selected for the generator and analyzer Function meter. To revert to the previous state use the command argument 0.

AP.Sweep.SinglePoint ❶❷ Property

Syntax `AP.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 (`AP.Sweep.Start`) the Data Editor will be automatically displayed and a single measurement taken at the Sweep Start value of Source 1.

See Also `AP.Sweep.Source1.Start`

❷ Example

```
Sub Main
   AP.File.OpenTest “SWEEPFFT.AT2"
   AP.Sweep.Repeat = False
   AP.Sweep.SinglePoint = False
   AP.Sweep.Stereo = False
   AP.Sweep.Timeout(“sec”) = 3
   AP.Sweep.Source1.Id = 5051    ‘Set Source 1 Gen Freq
   AP.Sweep.Source1.LogLin = 1
   AP.Sweep.Source1.Start(“Hz”) = 20000
   AP.Sweep.Source1.Stop(“Hz”) = 20
   AP.Sweep.Source1.Steps = 15
```

```
    AP.Sweep.Source1.AutoDiv = False
    AP.Sweep.Source1.Div = 10
    AP.Sweep.Source2.Id = 5052 'Set Source 2 Gen Ampl A
    AP.Sweep.Source2.LogLin = 1
    AP.Sweep.Source2.Start("Vrms") = 5
    AP.Sweep.Source2.Steps = 2
    AP.Sweep.Source2.Stop("Vrms") = 1
    AP.Sweep.Start
End Sub
```

AP.Sweep.Source1.AutoDiv

❶❷ Property

Syntax `AP.Sweep.Source1.AutoDiv`

Data Type Boolean

True	automatically select the number of divisions.
False	Use the number of divisions defined by the AP.Sweep.Source1.Div command.

Description This command enables or disables automatic selection of the number of linear horizontal axis divisions displayed for Source 1 sweeps.

See Also `AP.Sweep.Source1.Div, AP.Sweep.Source1.LogLin`

Example See example for `AP.Sweep.SinglePoint`.

AP.Sweep.Source1.Div

❶❷ Property

Syntax `AP.Sweep.Source1.Div`

Data Type Long Number of divisions displayed.

Description This command sets the number of divisions that are to be displayed for a linear horizontal axis for a Source 1. The `AP.Sweep.Source1.AutoDiv` must be disabled.

See Also `AP.Sweep.Source1.AutoDiv, AP.Sweep.Source1.LogLin`

Example See example for `AP.Sweep.SinglePoint`.

AP.Sweep.Source1.EndOn ❶❷ Property

Syntax `AP.Sweep.Source1.EndOn(`*`unit$`*`)`

Data Type Double — Refer to the setting or reading defined by the `AP.Sweep.Source1.Id` command to determine the appropriate range of acceptable values.

Parameters

Name	Description
unit$	Refer to the setting or reading defined by the `AP.Sweep.Source1.Id` command to determine the appropriate unit selections.

Description

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.

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 APWIN software) is providing the signal. In these cases, APWIN software cannot control the values, direction of progression (high to low versus low to high), or dwell times of the signal. APWIN 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.

❷ **Example**

```
Sub Main
'This test requires an external sweep source.
   AP.File.OpenTest "SweepD.at2"
   AP.Sweep.Source1.Id = 5901'Set Source 1 to Anlr.FreqA
   AP.Sweep.Source1.EndOn("Hz") = 2500 'Set Sweep End _
      On to 2.5kHz
   AP.Sweep.Source1.MinLevelID = 5903 'Select _
      Anlr.LevelA for MinLevel
   AP.Sweep.Source1.MinLevel("dBu") = -40 'Set Min _
```

```
      Level to 100mV
   AP.Sweep.Source1.Spacing("%") = 3 'Set Spacing to 3%
   AP.Sweep.Start                'Wait for external sweep
End Sub
```

AP.Sweep.Source1.Id

❶❷ Property

Syntax `AP.Sweep.Source1.Id`

Data Type Long — Instrument Parameter ID#.

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 D to obtain instrument parameter identification numbers.

Example See example for `AP.Sweep.SinglePoint`.

AP.Sweep.Source1.LogLin

❶❷ Property

Syntax `AP.Sweep.Source1.LogLin`

Data Type Integer

0	Logarithmic horizontal axis and step type.
1	Linear horizontal axis and step type.

Description This command determines the Source 1 horizontal axis type and the sweep step type.

See Also `AP.Sweep.Source1.Div, AP.Sweep.Source1.AutoDiv`

Example See example for `AP.Sweep.SinglePoint`.

AP.Sweep.Source1.MinLevel

02 Property

Syntax `AP.Sweep.Source1.MinLevel(`*unit$*`)`

Data Type Double

Refer to the reading defined by the `AP.Sweep.Source1.MinLevelId` command to determine the appropriate range of acceptable values.

Parameters

Name	Description
unit$	Refer to the reading defined by the `AP.Sweep.Source1.MinLevelId` command to determine the appropriate unit selections.

Description

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.

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 APWIN software) is providing the signal. In these cases, APWIN software cannot control the values, direction of progression (high to low versus low to high), or dwell times of the signal. APWIN 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 `AP.Sweep.Data2.Id, AP.Sweep.MinLevelSource`

AP.Sweep.Source1.MinLevelId ❶❷ Property

Syntax `AP.Sweep.Source1.MinLevelId`

Data Type Long Instrument Parameter ID#.

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 D to obtain instrument parameter identification numbers.

AP.Sweep.Source1.Multiply ❶❷ Property

Syntax `AP.Sweep.Source1.Multiply`

Data Type Double

Description This command sets the Source 1 Log Sweep Multiply factor used to determine the next Source 1 sweep setting.

See Also `AP.Sweep.Source1.Start, AP.Sweep.Source1.Stop, AP.Sweep.Source1.LogLin, AP.Sweep.Source1.Steps`

AP.Sweep.Source1.Spacing ❶❷ Property

Syntax `AP.Sweep.Source1.Spacing(`*`unit$`*`)`

Data Type Double

Parameters

Name	Description
unit$	% 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 `AP.Sweep.Source1,EndOn`.

AP.Sweep.Source1.Start

12 Property

Syntax `AP.Sweep.Source1.Start(unit$)`

Data Type Double — Refer to the setting or reading defined by the `AP.Sweep.Source1.Id` command to determine the appropriate range of acceptable values.

Parameters

Name	Description
unit$	Refer to the setting or reading defined by the `AP.Sweep.Source1.Id` 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 1 and to be displayed on the graph horizontal axis. In the case of an external sweep (a reading selected at Source 1 instead of a setting), this value determines the graph horizontal axis end point and, in conjunction with the `AP.Sweep.Source1.Stop` command, defines the expected direction of change of the Source 1 parameter during the sweep.

See Also `AP.Sweep.Source1.Stop`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Source1.Steps

12 Property

Syntax `AP.Sweep.Source1.Steps`

Data Type Long

Description This command sets the number of Source 1 steps that a log or linear sweep makes between the Source 1 Start and Stop values.

See Also `AP.Sweep.Source1.Start, AP.Sweep.Source1.Stop, AP.Sweep.Source1.LogLin, AP.Sweep.Source1.StepSize`

Example See example for `AP.Sweep.SinglePoint`.

AP.Sweep.Source1.StepSize ❶❷ Property

Syntax `AP.Sweep.Source1.StepSize(`*`unit$`*`)`

Data Type Double Source 1 step size.

Description This command sets the Source 1 Linear Sweep Step Size used to determine the next Source 1 sweep setting.

Name	Description
unit	Refer to the setting or reading defined by the `AP.Sweep.Source1.Id` command to determine the appropriate unit selections.

See Also `AP.Sweep.Source1.Start, AP.Sweep.Source1.Stop, AP.Sweep.Source1.LogLin, AP.Sweep.Source1.Steps`

AP.Sweep.Source1.Stop ❶❷ Property

Syntax `AP.Sweep.Source1.Stop(`*`unit$`*`)`

Data Type Double Refer to the setting or reading defined by the `AP.Sweep.Source1.Id` command to determine the appropriate range of acceptable values.

Parameters

Name	Description
unit$	Refer to the setting or reading defined by the `AP.Sweep.Source1.Id` 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 1 and to be displayed on the graph horizontal axis. In the case of an external sweep (a reading selected at Source 1 instead of a setting), this value determines the graph horizontal axis end point and, in conjunction with the `AP.Sweep.Source1.Start` command, defines the expected direction of change of the Source 1 parameter during the sweep.

See Also `AP.Sweep.Source1.Start`

Example See example for `AP.Sweep.Append`.

AP.Sweep.Source1.Table

❶❷ Method

Syntax `AP.Sweep.Source1.Table(filename$, column%)`

Parameters

Name	Description
filename$	Any valid DOS path and file name. The file must be an APWIN sweep file (.ads). Enter "None" for the file name to remove the sweep file from Source1.
column%	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 successfull.
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.

AP.Sweep.Source2.Id

❶❷ Property

Syntax `AP.Sweep.Source2.Id`

Data Type Long Instrument Parameter ID#.

Description This command is used to select the instrument parameter, which will define settings for Source 2.

Refer to Appendix D to obtain instrument parameter identification numbers.

Example See example for `AP.Sweep.SinglePoint`.

AP.Sweep.Source2.LogLin ❶❷ Property

Syntax `AP.Sweep.Source2.LogLin`

Data Type Integer

0	Logarithmic step type.
1	Linear step type.

Description This command determines if the sweep steps will be Logarithmically or linear spaced.

Example See example for `AP.Sweep.SinglePoint`.

AP.Sweep.Source2.Multiply ❶❷ Property

Syntax `AP.Sweep.Source2.Multiply`

Data Type Double

Description This command sets the Source 2 Log Sweep multiply factor used to determine the next Source 2 Sweep setting.

See Also `AP.Sweep.Source2.Start, AP.Sweep.Source2.Stop, AP.Sweep.Source2.LogLin, AP.Sweep.Source2.Steps`

AP.Sweep.Source2.Start ❶❷ Property

Syntax `AP.Sweep.Source2.Start(`*unit$*`)`

Data Type Double — Refer to the setting defined by the `AP.Sweep.Source2.Id` command to determine the appropriate range of acceptable values.

Parameters

Name	Description
unit$	Refer to the setting defined by the `AP.Sweep.Source2.Id` command to determine the appropriate range of acceptable values.

Description This command sets the first setting to be used in the Source 2 sweep.

See Also `AP.Sweep.Source2.Stop`

Example See example for `AP.Sweep.SinglePoint`.

AP.Sweep.Source2.Steps

❶❷ Property

Syntax **`AP.Sweep.Source2.Steps`**

Data Type Long

Description This command sets the number of Source 2 steps that a log or linear sweep makes between the Source 2 Start and Stop values.

See Also `AP.Sweep.Source2.Start, AP.Sweep.Source2.Stop, AP.Sweep.Source2.LogLin, AP.Sweep.Source2.StepSize`

Example See example for `AP.Sweep.SinglePoint`.

AP.Sweep.Source2.StepSize

❶❷ Property

Syntax **`AP.Sweep.Source2.StepSize(`*`unit$`*`)`**

Data Type Double — Refer to the setting defined by the `AP.Sweep.Source2.Id` command to determine the appropriate range of acceptable values.

Parameters

Name	Description
`unit$`	Refer to the setting defined by the `AP.Sweep.Source2.Id` command to determine the appropriate unit selections.

Description This command sets the Source 2 Linear Sweep Step Size used to determine the next Source 2 sweep setting.

See Also `AP.Sweep.Source2.Start, AP.Sweep.Source2.Stop, AP.Sweep.Source2.LogLin, AP.Sweep.Source2.Steps`

AP.Sweep.Source2.Stop ❶❷ Property

Syntax **AP.Sweep.Source2.Stop(*unit$*)**

Data Type Double — Refer to the setting defined by the AP.Sweep.Source2.Id command to determine the appropriate range of acceptable values.\

Parameters

Name	Description
unit$	Refer to the setting defined by the AP.Sweep.Source2.Id command to determine the appropriate unit selections.

Description This command sets the last setting to be used in the Source 2 sweep.

See Also AP.Sweep.Source2.Start

AP.Sweep.Start ❶❷ Method

Syntax **AP.Sweep.Start**

Data Type Boolean

True	Start Sweep (F9).
False	Terminate Sweep (ESC).

Description This command initiates or terminates a sweep.

Note: When using this command from an external application execution of additional commands will not be held off if the AP.Sweep.Repeat command is set to True. The AP.Sweep.Repeat command is also affected by the AP.Sweep.StartWithRepeat commmand.

See Also AP.Sweep.StartWithAppend, AP.Sweep.StartWithRepeat

❶ **Example**

```
Sub Main
   AP.File.OpenTest “FRQ-RESP.AT1"      ‘Open frequency _
      response test.
   AP.Sweep.Start                       ‘Start sweep.
   AP.File.SaveDataAs “FRQ-RESP.DAT”    ‘Save data.
```

41 Sweep

```
   AP.File.OpenTest "THD-FRQ.AT1"      'Open total _
      harmonic distortion + noise test.
   AP.Sweep.Start                      'Start sweep.
   AP.File.SaveDataAs "THD-FRQ.DAT"    'Save data.

   AP.File.OpenTest "RESIDNOI.AT1"     'Open residual _
      noise test.
   AP.Sweep.Start                      'Start sweep.
   AP.File.SaveDataAs "RESIDNOI.DAT"   'Save data.
End Sub
```

AP.Sweep.StartWithAppend ❶❷ Method

Syntax `AP.Sweep.StartWithAppend`

Data Type None

Description This command initiates a sweep in append mode which is equivalent to pressing the Ctrl+F9 function key.

See Also `AP.Sweep.Start, AP.Sweep.StartWithRepeat`

Example See example for `AP.Sweep.SinglePoint`.

AP.Sweep.StartWithRepeat ❶❷ Method

Syntax `AP.Sweep.StartWithRepeat`

Data Type None

Description This command initiates a sweep in repeat mode which is equivalent to pressing the Alt+F9 function key.

See Also `AP.Sweep.Start, AP.Sweep.StartWithAppend`

Example See example for `AP.Sweep.Repeat`.

AP.Sweep.Stereo

❶❷ Property

Syntax `AP.Sweep.Stereo`

Data Type Boolean

True	Enable Stereo Sweep
False	Disable Stereo Sweep.

Description This command enables or disables the stereo sweep feature on the Sweep panel.

Example See example for `AP.Sweep.SinglePoint`.

AP.Sweep.Timeout

Property

Syntax `AP.Sweep.Timeout(`*unit$*`)`

Data Type Double — Timeout values of 0 to 3000 seconds (50 minutes) are allowed.

Parameters

Name	Description
unit$	Sec unit only.

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 includes a line for each measured point which timed out durring the sweep resulting in a row showing the measured value and a letter T.

See Appendix C for Settling Algorithm and parameter name descriptions.

Example See example for `AP.Sweep.SinglePoint`.

User Notes

User Notes

User Notes

User Notes

Switcher

AP.SWR.ChABIn ❶❷ Property

Syntax `AP.SWR.ChABIn`

Data Type Long 0 - 192

Description This command sets the channel A and B connections of the Input switchers simultaneously. The channel A input is set to the specified channel number. The channel B input differs from the specified channel number by the value of the `AP.SWR.BOffset` command.

Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

See Also `AP.SWR.ChBOffset`

Example See example for `AP.SWR.Mode.`

AP.SWR.ChABInOut ❶❷ Property

Syntax `AP.SWR.ChABInOut`

Data Type Long 0 - 192

Description This command sets the channel A and B connections of the Input and Output switchers simultaneously. The channel A input is set to the specified channel number. The channel B input differs from the specified channel number by the value of the `AP.SWR.Boffset` command. The channel A output differs from the specified channel number by the value of the `AP.SWR.OutOffset` command. The channel B output differs from the specified channel number by the sum of the values of the AP.SWR.Boffset and the `AP.SWR.OutOffset` commands.

Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

See Also `AP.SWR.ChBOffset, AP.SWR.OutOffset`

Example See example for `AP.SWR.Mode.`

AP.SWR.ChABOut

12 Property

Syntax **`AP.SWR.ChABOut`**

Data Type Long 0 - 192

Description This command sets the channel A and B connections of the Output switchers simultaneously. The channel A output is set to the specified channel number. The channel B output differs from the specified channel number by the value of the `AP.SWR.ChBOffset` command.

Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

See Also `AP.SWR.ChBOffset`

Example See example for `AP.SWR.Mode.`

AP.SWR.ChAIn

12 Property

Syntax **`AP.SWR.ChAIn`**

Data Type Long 0 - 192

Description This command sets the switcher channel A input channel.

Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

Example See example for `AP.SWR.Mode.`

AP.SWR.ChAInOut

12 Property

Syntax **`AP.SWR.ChAInOut`**

Data Type Long

0 - 192

Description This command sets the channel A connections of the Input and Output switchers simultaneously. The channel A input is set to the specified channel number. The channel A output differs from the specified channel number by the value of the `AP.SWR.OutOffset` command.

Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

Example See example for `AP.SWR.Mode.`

AP.SWR.ChAOut ❶❷ Property

Syntax `AP.SWR.ChAOut`

Data Type Long 0 - 192

Description This command sets the switcher channel A Output channel.

Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

Example See example for `AP.SWR.Mode.`

AP.SWR.ChBIn ❶❷ Property

Syntax `AP.SWR.ChBIn`

Data Type Long 0 - 192

Description This command sets the switcher channel B Input channel.

Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

Example See example for `AP.SWR.Mode.`

AP.SWR.ChBInOut

12 Property

Syntax `AP.SWR.ChBInOut`

Data Type Long 0 - 192

Description This command sets the channel B connections of the Input and Output switchers simultaneously. The channel B input is set to the specified channel number. The channel B output differs from the specified channel number by the value of the `AP.SWR.OutOffset` command.

Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

See Also `AP.SWR.OutOffset`

Example See example for `AP.SWR.Mode.`

AP.SWR.ChBOffset

12 Property

Syntax `AP.SWR.ChBOffset`

Data Type Long 1 - 192

Description This command determines the channel number difference between channel B and the specified channel A.

See Also `AP.SWR.ChABIn, AP.SWR.ChABInOut, AP.SWR.ChABOut`

Example See example for `AP.SWR.Mode.`

AP.SWR.ChBOut

12 Property

Syntax `AP.SWR.ChBOut`

Data Type Long 0 - 192

Description This command sets the channel B output channel.

Channel numbers 1 to 192 are available, where 0 means all channels off. Any other number results in no action taken.

Example See example for `AP.SWR.Mode`.

AP.SWR.Mode ❶❷ Property

Syntax `AP.SWR.Mode`

Data Type Integer

0	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.
1	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.
2	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 ❷

```
Const INDEPENDENT As Integer = 0    'B independent of A
```

```
Const B_ONLY_A_OFF As Integer = 1   'All B on, All A off
Const COMPLEMENT As Integer = 2     'All B except _
   channel specified by A
Sub Main
   Dim switch As Integer, signal As Double, msg As String

   signal = 1.0                     'Use 1 V signal
   AP.Gen.ChAAmpl("V") = signal     'Set gen out level
   AP.Gen.Output = True             'Turn output on
   AP.SWR.Mode = INDEPENDENT        'Set Mode
   AP.SWR.OutOffset = 1             'ChAOut = ChAIn + 1
   AP.SWR.ChBOffset = 2             'ChB = ChA + 2
   For Switch = 1 To 6              'Sweep switches 1 to 6
      AP.SWR.ChAIn = Switch
      'AP.SWR.ChBIn = Switch        'Any of these switch
      'AP.SWR.ChABIn = Switch       ' commands can be used
      'AP.SWR.ChAOut = Switch       ' to sweep the channel
      'AP.SWR.ChBOut = Switch       ' A and/or B
      'AP.SWR.ChABOut = Switch      ' input and/or output
      'AP.SWR.ChAInOut = Switch     ' switches
      'AP.SWR.ChBInOut = Switch     ' with the appropriate
      'AP.SWR.ChABInOut = Switch    ' offsets
      AP.Anlr.ChALevelTrig
      While AP.Anlr.ChALevelReady = 0
      Wend
      rdg = AP.Anlr.ChALevelRdg("V")
      If rdg > 0.5 * signal Then 'any signal
         msg = msg & "Ch A In " & Switch & "<-> Ch A _
            Out " & Switch + AP.SWR.OutOffset & Chr(13)
      End If
   Next Switch
   AP.Prompt.Text = msg
   AP.Prompt.ShowWithContinue
   Stop
End Sub
```

AP.SWR.OutOffset

❶❷ Property

Syntax **AP.SWR.OutOffset**

Data Type Long 0 - 192

Description This command determines the channel number difference between the specified channel number and the output switcher channel.

See Also `AP.SWR.ChAInOut, AP.SWR.ChBInOut, AP.SWR.ChABInOut`

Example See example for `AP.SWR.Mode.`

User Notes

User Notes

User Notes

User Notes

User Notes

Sync/Ref Input

AP.Sync.DelayRdg ❷ Property

Syntax **AP.Sync.DelayRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description
unit$	The following units are available, sec.

Description This command returns a settled reading for the Sync Delay, In from Ref In field on the Sync/Ref Input panel. The reading is the time (phase) delay of the selected front panel XLR, BNC, or optical connector with respect to the selected rear panel AES/EBU Reference (sync) input signal. This feature is not relevant with general purpose serial or parallel formats.

Example

```
Const NOT_READY As Boolean = False
Const FLAT As Integer = 2
Const AES As Integer = 0
Const Z110 As Integer = 1

Sub Main
   Dim delay As Double

   AP.Application.NewTest      'Reset panels
   AP.Sync.SourceInput = AES   'Set Sync input source
   AP.Sync.Impedance = Z110    '110 ohm input impedance
   AP.Sync.Source = True       'Turn on source
   AP.Sync.DelaySettling 10e-3, 100e-9, "Sec", 3, 0.0, _
      NONE
   AP.Sync.DelayTrig           'Trigger a new reading
   While AP.Sync.DelayReady = NOT_READY 'wait for _
      reading to settle
   Wend
   delay = AP.Sync.DelayRdg("SEC")  'Measure the _
      sync-signal delay
'now that we have delay, use this to output test signal
```

```
    AP.Sync.Source = False      'Generate sync
    AP.Sync.OutDelay = True     'Enable output delay
    AP.Sync.OutDelayFromRef("SEC") = delay 'Match _
       measured output delay
    'now perform further testing on DUT ...
End Sub
```

AP.Sync.DelayReady

❷ Property

Syntax `AP.Sync.DelayReady`

Data Type Integer

0	Reading not ready.
>0	Reading ready.

Description This command returns the Sync In Delay settled 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 `AP.Sync.DelayRdg` command will zero the ready count.

If the reading is found to be ready, a call to the Frequency A 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 `AP.Sync.DelayTrig` command.

See Also `AP.Sync.DelayRdg, AP.Sync.DelaySettling, AP.Sync.DelayTrig`

Example See example for `AP.Sync.DelayRdg`.

AP.Sync.DelaySettling

❷ Method

Syntax **`AP.Sync.DelaySettling(`***`tolerance#, floor#, floorunit$, points%, delay#, algorithm%`***`)`**

Description This command sets the settling parameters for the `AP.Sync.DelaySettling` command.

See Appendix C for Settling Algorithm and parameter name descriptions.

See Also `AP.Sync.DelayRdg, AP.Sync.DelayReady, AP.Sync.DelayTrig`

Example See example for `AP.Sync.DelayRdg`.

AP.Sync.DelayTrig ❷ Method

Syntax **`AP.Sync.DelayTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.Sync.DelayRdg` command. The reading in progress is aborted.

See Also `AP.Sync.DelayRdg, AP.Sync.DelayReady, AP.Sync.DelaySettling`

Example See example for `AP.Sync.DelayRdg`.

AP.Sync.Freq ❷ Property

Syntax **`AP.Sync.Freq(`*`unit$`*`)`**

Data Type Double 8kHz - 54kHz

Parameters

Name	Description
unit$	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 System Two 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 AP.Sync.

Example

```
Const NTSC As Integer = 2
Const Z75 As Integer = 1

Sub Main
   AP.Sync.Source = NTSC      'Set Sync input source
   AP.Sync.Impedance = Z75    '75 ohm input impedance
   AP.Sync.Source = False     'Turn off sourcing

   rdg = AP.Sync.Freq("Hz")   'get input sync frequency
   If (rdg < lower_limit) Or (rdg > upper limit) Then
      'input sync freq not close enough, flag an error _
         and ...
      End
   Else
      If AP.Sync.OutRangeRdg Or AP.Sync.UnLockedRdg Then
         'internal clock not sync'ed, flag an error _
            and ...
         End
      Else
         AP.Sync.Source = True   'Turn on sourcing
         'now perform further testing on DUT...
      End If
   End If
End Sub
```

AP.Sync.FreqRdg

❷ Property

Syntax **AP.Sync.FreqRdg(*unit$*)**

Data Type Variant

Parameters

Part	Description

unit$	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 System Two (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 `AP.Sync.FreqReady, AP.Sync.FreqTrig`

Example

```
Sub Main
   AP.Sync.Source = False  'Freq rdg only w/src off
   AP.Sync.FreqTrig        'Start a new reading
   While AP.Sync.FreqReady = False  'Wait for reading
      'do other tasks while waiting for reading ...
   Wend
   reading1 = AP.Sync.FreqRdg("Hz")
   Debug.Print "Sync Input Frequency = "; _
      Format(Reading1, "#.0000");" Hz"
End Sub
```

Example Output

```
Sync Input Frequency = 48000.0017 Hz
```

AP.Sync.FreqReady ❷ Property

Syntax `AP.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 `AP.Sync.FreqRdg` command will zero the ready count.

If the reading is found to be ready, a call to the `AP.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 `AP.Sync.FreqTrig` command.

See Also `AP.Sync.FreqRdg, AP.Sync.FreqTrig`

Example See example for `AP.Sync.FreqRdg`.

AP.Sync.FreqTrig ❷ Method

Syntax **`AP.Sync.FreqTrig`**

Description Causes a restart of the reading cycle and zeros the ready count for the `AP.Sync.FreqRdg` command. The reading in progress is aborted.

See Also `AP.Sync.FreqRdg, AP.Sync.FreqReady`

Example See example for `AP.Sync.FreqRdg`.

AP.Sync.Impedance ❷ Property

Syntax **`AP.Sync.Impedance`**

Data Type Integer

The following list contains the selections relevant to the `AP.Sync.Source` command for the AES Sync Rate selection.

0 Hi Impedance

1	110 Ohms

The following list contains the selections relevant to the `AP.Sync.Source` command for the Squarewave, NTSC Video Sync Horz Rate, PAL / SECAM Video Sync Horz Rate selections.

0	Hi Impedance
1	75 Ohms

Description This command controls the input impedance for Balanced and Un-Balanced Sync Input configurations.

See Also `AP.Sync.Source`

Example See example for `AP.Sync.DelayRdg`.

AP.Sync.OutDelay ❷ Property

Syntax **`AP.Sync.OutDelay`**

Data Type Boolean

True	Enable delay.
False	Disable delay.

Description This command enables or disables the specified generator output delay relative to the Ref Our as defined by the `AP.Sync.OutDelayFromRef` command. When delay is not required, this feature should be disabled rather than set the `AP.Sync.OutDelayFromRef` command to a zero value, since residual jitter is slightly higher when the output delay feature is enabled.

The ON/OFF button at the left of the Source selection field connects the selected source signal to System Twos internal phase locked loops. The Input Frequency field will be blanked when the switch is turned on, since the reading will be redundant if lock is achieved and will be incorrect if lock is not possible.

See Also `AP.Sync.OutDelayFromRef`

Example See example for `AP.Sync.DelayRdg`.

AP.Sync.OutDelayFromRef ❷ Property

Syntax `AP.Sync.OutDelayFromRef(unit$)`

Data Type Double -10.42 to 10.34 sec

Parameters

Name	Description
unit$	String that designates the desired unit. The following units are valid for this command: UI, sec

Description This command controls the time (phase) delay of the Digital Generator (front panel) output relative to the rear panel AES/EBU REF OUT XLR connector. To use this feature, the Ref Out signal would be connected to a digital device under test as house sync while System Twos Digital Generator drives the devices digital signal input. The devices tolerance to delay from reference may then be tested by entering different values into the Output Delay from Ref value. When delay is not required, this feature should be turned off via the `AP.Sync.OutDelayFromRef` command or manualy via the On/Off button at the right of the field rather than set to a zero value with the button On, since residual jitter is slightly higher when the output delay feature is on.

Example See example for `AP.Sync.DelayRdg`.

AP.Sync.OutOfRangeRdg ❷ Property

Syntax `AP.Sync.OutOfRangeRdg`

Parameters None

Result Boolean

True	In Range
False	Out Of Range

Description This command returns a unsettled reading for the Sync Out Of Range indicator.

See Also `AP.Sync.UnLockedRdg`

Example See example for `AP.Sync.Freq`.

AP.Sync.Source ❷ Property

Syntax `AP.Sync.Source`

Data Type Boolean

True	Enable.
False	Disable.

Description This command enables or disables the external sync input.

See Also `AP.Sync.SourceInput`

Example See example for `AP.Sync.FreqRdg`.

AP.Sync.SourceInput ❷ Property

Syntax `AP.Sync.SourceInput`

Data Type Integer

0	AES Sync Rate:
1	Squarewave:
2	NTSC Video Sync Horz Rate:
3	PAL / SECAM Video Sync Horz Rate:

Description This command sets the input type for the external sync input.

See Also `AP.Sync.Source`

Example See example for `AP.Sync.DelayRdg`.

AP.Sync.UnLockedRdg ❷ Property

Syntax `AP.Sync.UnLockedRdg`

Parameters None

Result Boolean

True	Locked

False	UnLocked

Description This command returns a unsettled reading for the Sync Un Locked indicator.

See Also `AP.Sync.OutOfRangeRdg`

Example See example for `AP.Sync.Freq`.

User Notes

User Notes

User Notes

Appendix C 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 Source parameter permitted to proceed to the next step.

Settling Parameter Discriptions

Name	Discription
Tolerance#	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 APWIN 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. Acquistion 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% 0 = 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 System Two 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.

1 = 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.

2 = 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.

3 = 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 D Sweepable 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 AP.Sweep.????.ID command designating the desired sweep parameter to be changed.

Unit	Sweep panel ID Text	Value
❶❷	Anlr.Ampl	5906
❶	Anlr.Ampl (2Ch)	5915
❶❷	Anlr.Bandpass	5907
❶❷	Anlr.BandReject	5908
❶❷	Anlr.BP Ampl	5917
❶❷	Anlr.BP Pct	5918
❶❷	Anlr.BPBR Freq	5155
❶❷	Anlr.CCIF	5912
❶❷	Anlr.DIM	5913
❷	Anlr.Freq A	5901
❷	Anlr.Freq A & Freq B	5920
❷	Anlr.Freq B	5902
❷	Anlr.Level A	5903
❷	Anlr.Level A & Level B	5919
❷	Anlr.Level B	5904
❶❷	Anlr.Pct	5916

❶❷	Anlr.Phase	5905
❶❷	Anlr.SMPTE	5911
❶❷	Anlr.THD Ampl	5909
❶❷	Anlr.THD Pct	5910
❶❷	Anlr.WF	5914
❶	Bittest.Ch.A Errors	6051
❶	Bittest.Ch.A Data	6049
❶	Bittest.Ch.B Errors	6052
❶	Bittest.Ch.B Data	6050
❶	Bittest.DGen Ampl	5599
❶	Bittest.DGen Freq	5598
❶	Bittest.DGen Value	5600
❶	Codec.Ch.1 Ampl	6333
❶	Codec.Ch.1 Phase	6041
❶	Codec.Ch.1-2 Phase	6042
❶	Codec.Ch.2 Ampl	6336
❶	Codec.DGen Ampl	5571
❶	Codec.FFT Freq	5630
❶	Codec.FFT Time	5631
❶	Codec.Freq Resolution	5570
❶❷	Dcx.DC Out 1	5258
❶❷	Dcx.DC Out 2	5260
❶❷	Dcx.Dig In	5953
❶❷	Dcx.Dig Out	5265
❶❷	Dcx.DMM Ohms	5952
❶❷	Dcx.DMM Volts	5951

❶❷	Dcx.Gate Delay	5271
❶❷	Dcx.Port A	5268
❶❷	Dcx.Port B	5269
❶❷	Dcx.Port C	5270
❷	DGen.Ampl A	5106
❷	DGen.Ampl A & Ampl B	5121
❷	DGen.Ampl B	5107
❷	DGen.Ampl Ratio	5105
❷	DGen.Center Freq	5134
❷	DGen.Ch. A Freq	5114
❷	DGen.Ch. B Freq	5115
❷	DGen.Freq	5102
❷	DGen.High Freq	5133
❷	DGen.IM Freq	5104
❷	DGen.Samples/Step	5135
❷	Dio.Common Mode Ampl	5317
❷	Dio.Common Mode Freq	5318
❷	Dio.Delay from Output	6104
❷	Dio.Input Resolution	5325
❷	Dio.Input Sample Rate	6101
❷	Dio.Input Voltage	6102
❷	Dio.Interface Jitter	6105
❷	Dio.Interfering Noise Ampl	5305
❷	Dio.Jitter Freq	5322
❶	Dio.Output Res.	5333
❷	Dio.Output Resolution	5326

❷	Dio.Output Sample Rate	5301
❷	Dio.Output Voltage	5304
❷	Dio.Peak Monitor A	6121
❷	Dio.Peak Monitor B	6122
❶	Dio Receive Sync	6117
❶	Dio.Transmit Sync	6118
❷	DSP Audio Anlr.Ampl	6014
❷	DSP Audio Anlr.Bandpass	6019
❷	DSP Audio Anlr.BP/BR Filter Freq	5542
❷	DSP Audio Anlr.Crosstalk	6016
❷	DSP Audio Anlr.Freq A	6009
❷	DSP Audio Anlr.Freq B	6010
❷	DSP Audio Anlr.Level A	6005
❷	DSP Audio Anlr.Level B	6006
❷	DSP Audio Anlr.Noise	6020
❷	DSP Audio Anlr.Ratio	6015
❷	DSP Audio Anlr.THD+N Ampl	6018
❷	DSP Audio Anlr.THD+N Ratio	6017
❷	Fasttest.Ch. 1 Jitter	6063
❷	Fasttest.Ch. 2 Jitter	6064
❶❷	Fasttest.Ch.1 Ampl	6309
❶❷	Fasttest.Ch.1 Phase	6033
❶❷	Fasttest.Ch.2 Ampl	6312
❶❷	Fasttest.Ch.2 Phase	6034
❶	Fasttest.DGen Ampl	5552
❶	Fasttest.FFT Freq	5626

❷	Fasttest.FFT Freq	5621
❶	Fasttest.FFT Time	5627
❷	Fasttest.FFT Time	5620
❶❷	Fasttest.Freq Resolution	5551
❶	Fasttrig.Ch.1 Ampl	6317
❶	Fasttrig.Ch.1 Phase	6037
❶	Fasttrig.Ch.1-2 Phase	6038
❶	Fasttrig.Ch.2 Ampl	6320
❶	Fasttrig.DGen Ampl	5561
❶	Fasttrig.FFT Freq	5628
❶	Fasttrig.FFT Time	5629
❶	Fasttrig.Freq Resolution	5560
❷	Fft.Ch.1 Ampl	6023
❷	Fft.Ch.2 Ampl	6026
❷	Fft.FFT Freq	5515
❷	Fft.FFT Pre-Trig Time	5519
❷	Fft.FFT Start Time	5518
❷	Fft.FFT Time	5516
❶	Fftgen.Ch.1 Ampl	6023
❶	Fftgen.Ch.2 Ampl	6026
❶	Fftgen.DGen Ampl	5514
❶	Fftgen.DGen Freq	5513
❶	Fftgen.FFT Freq.	5515
❶	Fftgen.FFT Time	5516
❶	Fftslide.Ch.1 Ampl	6301
❶	Fftslide.Ch.2 Ampl	6304

❶	Fftslide.FFT Freq	5528
❶	Fftslide.FFT Start Time	5526
❶	Fftslide.FFT Time	5529
❶	Fftslide.Pre-Trig Time	5527
❶	Gen.Ampl	5075
❷	Gen.Ampl A	5052
❷	Gen.Ampl B	5053
❷	Gen.Ampl A & Ampl B	5076
❷	Gen.Ampl Ratio	5086
❷	Gen.Burst Interval	5069
❷	Gen Burst On	5068
❷	Gen.Center Freq	5088
❷	Gen.Dual Ampl Ratio	5085
❶❷	Gen.Freq	5051
❶	Genanlr.2 Channel	6014
❶	Genanlr.DGen Ampl	5541
❶	Genanlr.DGen Freq	5540
❶	Genanlr.Freq A	6009
❶	Genanlr.Freq B	6010
❶	Genanlr.Level A	6005
❶	Genanlr.Level B	6006
❶	Harmonic.Filter Freq	6048
❶	Harmonic.Filter Freq	5590
❶	Harmonic.Filtered Ampl	6047
❶	Harmonic.Freq Offset	5592
❶	Harmonic.Harmonic	5591

❷	Intervu.Amplitude	6053
❷	Intervu.Freq	5613
❷	Intervu.Jitter	6055
❷	Intervu.Probability	6054
❷	Intervu.Time	5612
❶❷	Mls.Ch.1 Ampl	6325
❶❷	Mls.Ch.1 Phase	6045
❶❷	Mls.Ch.2 Ampl	6328
❶❷	Mls.Ch.2 Phase	6046
❶❷	Mls.DGen Ampl	5580
❶❷	Mls.MLS Freq	5581
❶❷	Mls.Ref Time	5579
❶❷	None	5049
❶❷	Swr.Ch. A Input	5201
❶❷	Swr.Ch. A Input/Output	5206
❶❷	Swr.Ch. A Output	5203
❶❷	Swr.Ch. A+B Input	5208
❶❷	Swr.Ch. A+B Input/Output	5210
❶❷	Swr.Ch. A+B Output	5209
❶❷	Swr.Ch. B Input	5202
❶❷	Swr.Ch. B Input/Output	5207
❶❷	Swr.Ch. B Output	5204
❶❷	Sync/Ref.In from Ref In Delay	6103
❶❷	Sync/Ref.Input Freq	6106
❶❷	Time.External Sweep Time	6253
❶❷	Time.Time Since Test Loaded	6251

Appendix E FFT Window Descriptions

Window	Discription
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.

Index

Audio precision

Audio Precision, Inc.
PO Box 2209
Beaverton, Oregon 97075-2209
U.S. Toll Free: 1-800-231-7350
Tel: (503) 627-0832 Fax: (503) 641-8906
email: techsupport@audioprecision.com
Web: www.audioprecision.com

Audio Precision
PO Box 2209
Beaverton, Oregon 97075-2209
Tel 503 627-0832 Fax 503 641-8906
US Toll Free 1-800-231-7350
email techsupport audioprecision.com
Web www.audioprecision.com