APWIN 2.22 Addendum

APWIN version 2.22 Addendum for
System Two Cascade and Cascade Plus
2.0 User's & Basic Extension Reference Manuals

APWIN Version 2.22 Addendum to Version 2 User's Manuals

...including new features for System Two Cascade *Plus*

This document is designed to be used in conjunction with and as a supplement to the existing APWIN version 2 manual sets.

September 2002

Audio Precision part number 8211.0141 Revision 0

Version 2.22 September 2002

Audio Precision
5750 SW Arctic Drive
Beaverton, Oregon 97005
Tel: 503-627-0832 Fax: 503-641-8906
US Toll Free: 1-800-231-7350
email: info@audioprecision.com

Printed in the United States of America

II0912143854

Contents

Chapter 1: S2C+

New Features Available Only in System Two Cascade Plus

This addendum discusses the new features and changes brought to the operation of System Two Cascade and Cascade *Plus* since the release of Audio Precision® APWIN™ version 2.0. Some of the features apply to new capabilities only available in the System Two Cascade *Plus* hardware and will not function without that instrument.

V 2.22

There have been four releases since APWIN version 2.0: Versions 2.11, 2.14, 2.2 and the current 2.22. The version icon (shown at right) indicates the version number associated with the new or changed feature described in the accompanying text.

See Chapter 2 for documentation of other changes and improvements for both System Two Cascade and Cascade *Plus* families since APWIN version 2.0.

FFT Spectrum Analyzer

There are two improvements to the Digital Analyzer's FFT Spectrum Analyzer instrument for System Two Cascade *Plus*.

4M FFT Acquisition Memory

V 2.2

The System Two Cascade *Plus* hardware provides significantly more memory for FFT signal acquisition, raising the maximum from 256 kBytes to 4 MBytes.

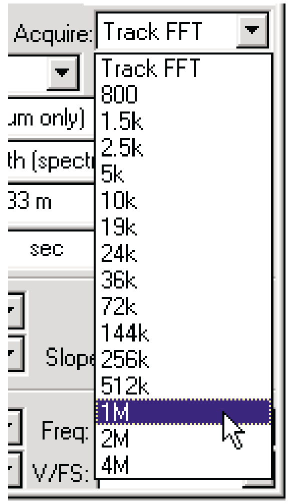


Figure 1. The FFT Acquisition memory list.

To access the additional memory, drop the Acquire list on the FFT Analyzer panel. Choose one of the following new options from the end of the list:

- **512k**
- **1M**
- **2M**
- **4M**

V 2.2

FFT Smoothing

A new display method called *Octave Smoothing* has been added to the FFT Analyzer for System Two Cascade *Plus:*

Previous versions of APWIN offered the FFT display options **Interpolate, Display Samples, Peak Values** and **Absolute Values** in the **Wave Display** field. These display methods only affect the time-domain, “oscilloscope” waveform view of the acquired data.

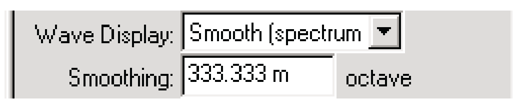


Figure 2. FFT Smoothing.

Now you can also select FFT octave smoothing by clicking the new fifth option in the **Wave Display** list, **Smooth (spectrum only)**. The **Smooth** display method only affects the frequency-domain, spectrum analyzer view of the acquired data.

Unlike FFT power averaging, which takes the average of a number of measurements, FFT smoothing is a display method that shows the results of one measurement as modified by a smoothing algorithm.

When **Smooth** is chosen the **Smoothing** setting field becomes available just under the **Wave Display** field. Specify the degree of smoothing you desire by entering a value between 0 to 2.64 octaves.

Octave smoothing is a common technique in loudspeaker response measurement, useful in revealing trends by smoothing out anomalies in the response curve. The APWIN implementation uses a hybrid FFT bin averaging and interpolation technique to achieve smooth results even at very low bin densities. Smoothing effectively passes the raw frequency-domain response data through multiple constant-Q bandpass filters, one filter centered on each frequency requested from the Sweep panel. The bandwidth of these filters, in octaves, is specified in the **Smoothing** field.

Chapter 2: S2C, S2C+

Changes Affecting
■ *System Two Cascade*
■ *System Two Cascade* Plus

New PSIA Panels **V 2.22**

Figure 3. The PSIA-2722 Programmable Serial Interface Adapter.

APWIN version 2.22 includes two new PSIA panels to control a new auxiliary unit, the PSIA-2722 Programmable Serial Interface Adapter. The PSIA-2722 is an accessory unit for System Two Cascade and Cascade *Plus* running under APWIN version 2.22.

Background: Serial Digital Interface

A serial interface adapter such as PSIA-2722 is required to transmit or receive digital signals and associated clock inputs and outputs for the non-AES3/IEC60958 serial digital audio formats often encountered in telecommunications and in converter design and testing. The settings necessary to configure the PSIA are easily accomplished in software, and converter-specific setups can be saved, reloaded or downloaded from the Web.

The PSIA panels

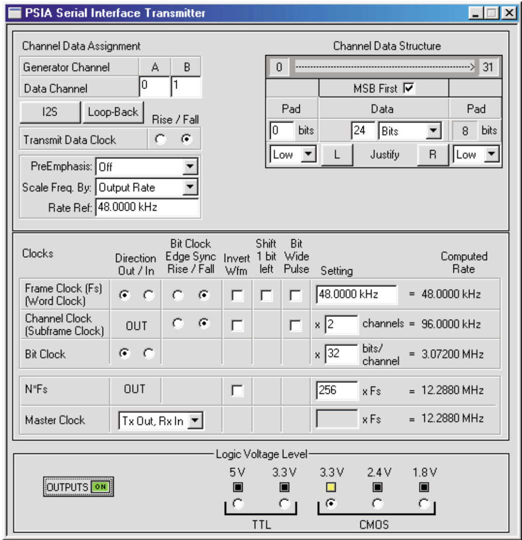


Figure 5. The PSIA Transmitter panel.

The APWIN PSIA panels are available by selecting **Panels > PSIA > Transmitter** or **Receiver** from the Menu bar, or by clicking the **PSIA Transmitter** or **PSIA Receiver** buttons on the toolbar. You can also launch the PSIA panels using the keyboard combinations **Ctrl+R** or **Ctrl+T.**

The top section of each panel controls the serial data configuration; below that is a matrix of settings and displays for the serial clocks; at the bottom of the panels are controls for the hardware interface voltages.

These panels have no function when a PSIA-2722 is not connected to the Cascade. See the documentation provided with PSIA-2722 for detailed information on the interconnection and use of the Programmable Serial Interface Adapter and the PSIA panels.

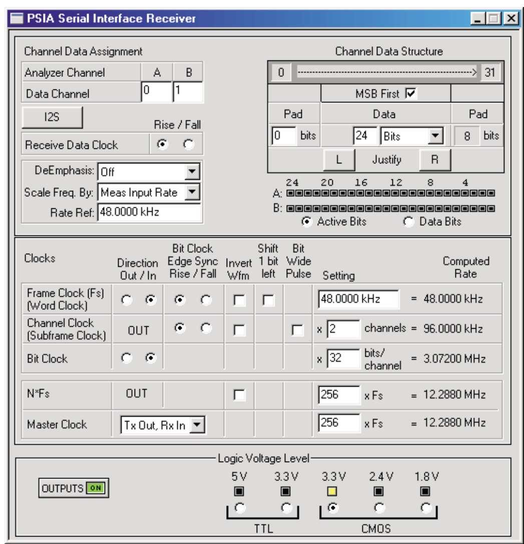


Figure 4. The PSIA Receiver panel.

DCX-127 Port D (J141) now addressable

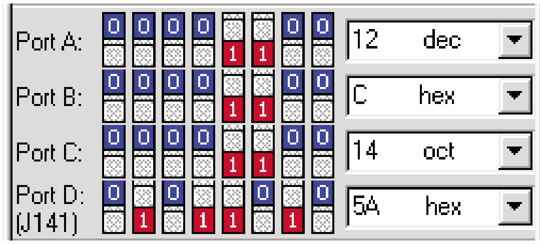


Figure 6. Port control on DCX panel.

Port D, the fourth Auxiliary Output port on the DCX-127 Multifunction Module, was previously only addressable by AP Basic command. APWIN Version 2.22 enables you to address this port directly from within the control software. Port D is available on the 15-pin D-Sub connector labeled “J141” on the DCX-127 rear panel.

New Graph Legend features

V 2.22

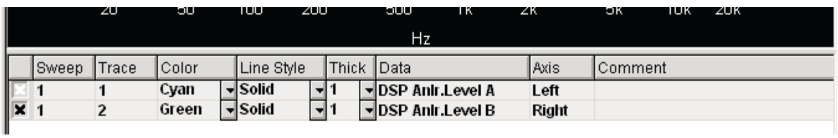


Figure 7. Graph Legend new columns.

Three new columns have been added to the Graph Legend window, displaying more information about each trace in an APWIN graph. The new columns are **Sweep**, **Trace** and **Comment**.

- In a multiple-sweep graph (such as appended sweeps or nested sweeps), **Sweep** identifies each by number.
- In a multiple-trace sweep (**Data 2** through **Data 6** used), **Trace** identifies each by number.

Comment provides a cell to enter an optional comment for each trace row in the Graph Legend.

V 2.22 Data Export as Excel spreadsheet now available

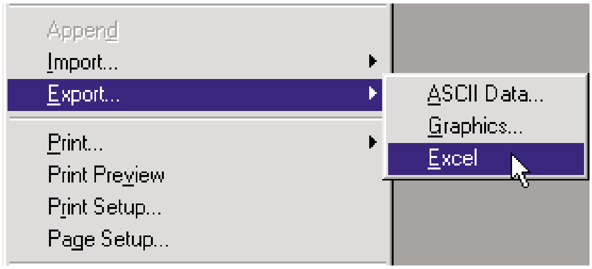


*Figure 9. New **File > Export > Excel** option.*

In addition to the ASCII text format available in previous versions, in APWIN 2.22 you can now export test data as a Microsoft Excel spreadsheet.

Digital I/O Panel

There are several changes to the Digital Input/Output panel which affect System Two Cascade.

V 2.22 PSIA Input and Output Format Selection

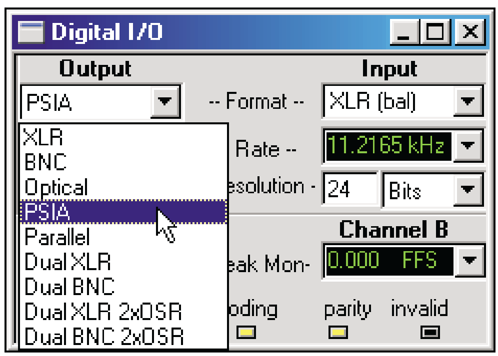


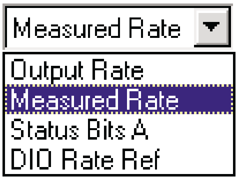
Figure 8. DIO Output: Format list.

The DIO **Input: Format** and **Output: Format** lists each now include a **PSIA** choice. The **PSIA** selections use the System Two Cascade parallel ports, but differ from the **Parallel** selections by enabling PSIA panel operation. **PSIA** may be selected independently for **Input** or **Output**.

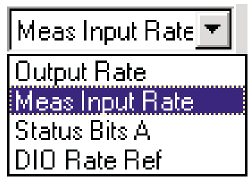
Digital Input Sample Rate Scaling

V 2.11

The choices available for the **Scale Freq. by** field on the Input side of the Digital I/O panel have changed.



Old Names



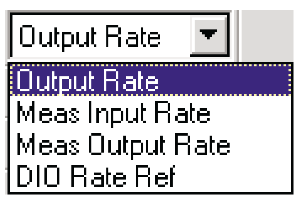
New Names

The function for each choice is the same as in earlier versions of APWIN. The list choice names have been changed to avoid confusion with choices on the new output **Scale Freq. by** list.

Digital Output Sample Rate Scaling

V 2.11

Previously, the frequency of the audio signal embedded in the digital output was scaled based solely on the setting in the **Sample Rate-OSR** field. A new **Scale Freq. by** field has been added to the Output side of the Digital I/O panel with these four options for digital output audio scaling:



- **Output Rate**, the default, selects the value entered in the **Sample Rate-OSR** field higher on the panel. This provides the same operation as in previous versions of APWIN.

  *The Parallel Digital Output was an exception to this rule in previous software versions. In APWIN versions 1.60 through 1.62, the embedded audio frequency was scaled by the value in the **Sample Rate-OSR** field; in APWIN versions 2.00 through 2.03 it was scaled by **DIO Rate Ref.***

- **Meas Input Rate** selects the value measured at the digital input port. This is the same value selected by the **Meas Input Rate** choice on the Input **Scale Freq. by** field.

- **Meas Output Rate** selects the value measured at the parallel output port when **Output Format** is set to **Parallel**. This value is the sample rate read from the DUT. **Meas Output Rate** defaults to **Sample Rate-OSR** (the same as the **Output Rate** choice) if an output other than parallel is used.
- **DIO Rate Ref** selects the value entered in the **Rate Ref** field, located on the Input side of the panel.

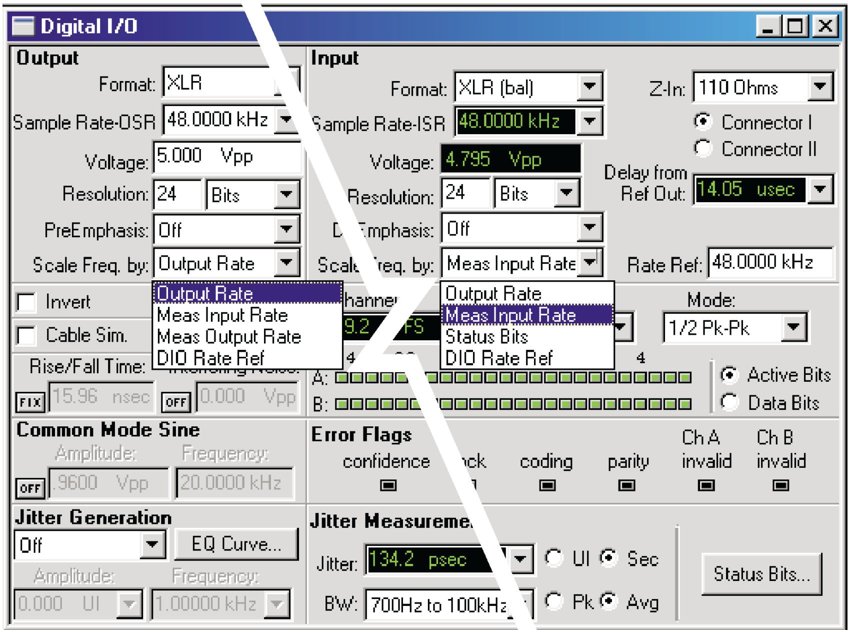


*Figure 10. New **Scale Freq by** features in Digital I/O Panel Output and Input sections.*

V 2.11 Parallel Input Sample Rate

Starting with APWIN 2.11, the parallel digital input sample rate is now correctly measured. Prior to version 2.11, the measured **Sample Rate-ISR** reading displayed incorrect values when the input **Format** was set to **Parallel**. The valid range for the measured parallel input sample rate is greater than 7001 Hz and less than 216 kHz.

V 2.11 μ-Law/A-Law

The International Telecommunication Union (ITU) specifies in its standard G.711 two similar approaches for reducing the bit rate in digital voice telephony. Called *μ-Law* and *A-Law*, these techniques have been widely used in digital telecommunications for a number of years.

μ-Law and A-Law are known as *companders*, converting 14-bit linear PCM samples (μ-Law) or 13-bit linear PCM samples (A-Law) to an 8-bit pseudo floating-point representation via *compression* at the encoder and *expansion* at the decoder.

A new feature as of version 2.11 enables APWIN to both transmit and receive μ-Law and A-Law encoded digital signals. Although most μ-Law and A-Law testing will be done via the parallel output and input ports in conjunction with a Serial Interface Adapter (Audio Precision PSIA-2722 or SIA-2322), these features support the AES3, S/PDIF and optical inputs and outputs as well, when the sample rate specified is above the minimum for these ports.

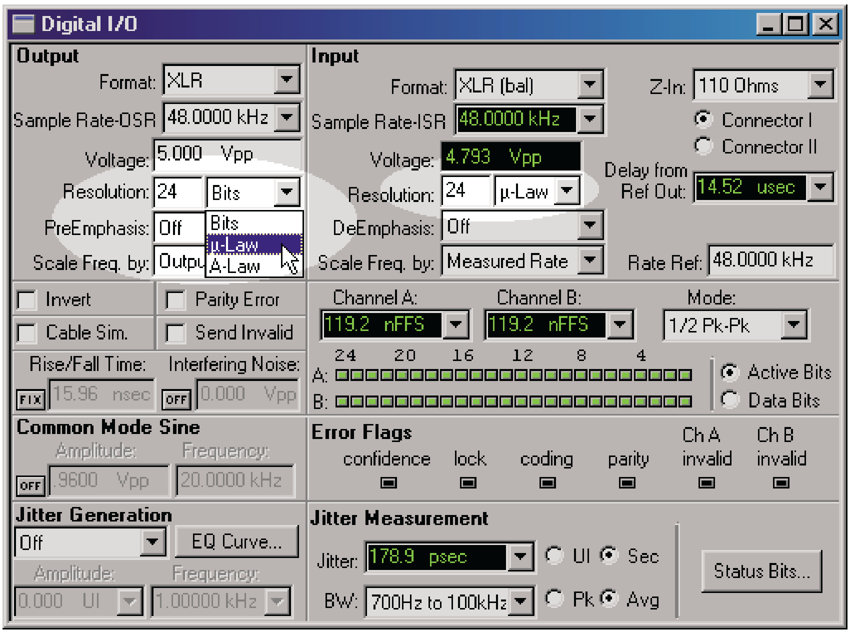


Figure 11. Digital I/O Panel showing µ-Law/A-Law settings.

μ-Law and A-Law can be selected by new control fields on the Digital I/O panel. The μ-Law/A-Law encoders are found on the Output side of the panel; the decoders are on the Input side. In either section, go just to the right of the **Resolution** field and click on the list arrow next to **Bits** setting. Then simply select **μ-Law** or **A-Law** from the list.

*The Digital Generator outputs a 14-bit or 13-bit signal when the **Resolution** is set to **µ-Law** or **A-Law**. Dither, if enabled, is properly scaled to these truncated word lengths.*

V 2.11

User Downloadable Filters

APWIN has a number of built-in filters implemented in software for the Digital Analyzer program, including two low-pass filters, three high-pass filters and six weighting filters. The use of these filters is discussed in detail in the *APWIN User's Manual for System Two Cascade Version 2* in pages 11-12 through 11-16.

APWIN version 2.11 for System Two Cascade added a powerful new feature to the Digital Analyzer: the ability to use custom-designed software filters in any of the three filter groups. These are called User Downloadable Filters.

Creating User Downloadable Filters

An APWIN User Downloadable Filter consists of a properly formatted text file containing information defining the characteristics of the filter. The filter must be named with one of these file name extensions:

- for a low-pass filter, *.afl;
- for a high-pass filter, *.afh;
- for a weighting filter, *.afw.

As long as the file meets the specifications set out below, you can create the filter by any means, including the use of a text editor or third-party software that has digital filter design capabilities.

User Downloadable Filter Design Constraints:

- Each file specifies a single infinite impulse response (IIR) filter at one or more sample rates.

Each IIR filter can be implemented by cascading one or more second-order sections, shown here:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} .$$

First-order sections are not allowed; a first-order section can be implemented with the second-order section by simply setting b_2 and a_2 to zero.

- A low-pass filter may not have more than 6 poles or 6 zeros; that is, a low-pass filter can have at most three second-order sections.
- A high-pass filter may not have more than 4 poles or 4 zeros; that is, a high-pass filter can have at most two second-order sections.

- A weighting filter may not have more than 8 poles or 8 zeros; that is, a weighting filter can have at most four second-order sections.
- Unstable filters are not allowed. An unstable filter is a filter with poles on or outside the unit circle on the *z*-plane.
- Filters with a gain of zero are not allowed. Such filters would result in all zero outputs.
- No coefficient may be greater than 2.0 or less than –2.0.

A simple text file format is used for each user-designed filter. The file format is specified by the following rules:

- All lines starting with # are comment lines.
- Blank lines, that is, lines consisting of zero or more spaces and tabs, are ignored.
- The parameters for filters are specified in a line-oriented fashion, with one entity specified per line.
- Each line has three components: a predefined *keyword*, followed by a *colon*, followed by the *keyword-dependent data*. Any white space surrounding these components is ignored. Three keywords are currently defined: **info**, **sample_rate**, and **biquad**.
- The **info** keyword specifies a text string of printable ASCII characters to be displayed on the APWIN User Downloadable Filters panel when the "**Filter Info**" box is clicked. The **info** comment must be fewer than 1024 characters in length. Only the first **info** keyword found is used; any others are ignored.
- A filter at a given sample rate is specified by the **sample_rate** keyword line followed by one or more **biquad** keyword lines.
- The **sample_rate** keyword is followed by a colon and a floating-point number expressing the sample rate in hertz. There is no limit on the number of sample rates. APWIN's built-in filters are provided at these sample rates:

8000 Hz	11025 Hz	12000 Hz	16000 Hz
22050 Hz	24000 Hz	32000 Hz	44100 Hz
48000 Hz	65536 Hz	88200 Hz	96000 Hz
131072 Hz	176400 Hz	192000 Hz	262144 Hz

Table 1. Sample rates for APWIN built-in filters.

- The **biquad** keyword is followed by a colon and five floating-point coefficients a_1, a_2, b_1, b_2, and b_0 in that order, as described in the equation above.

A valid sample file is shown here:

```
info: 100 Hz Butterworth high-pass filter

sample_rate: 8000.0
biquad: -1.8590763 0.8648249 -1.8048889 0.9024445 0.9024445
biquad: -1.9357148 0.9417005 -2.0000000 1.0000000 1.0000000

sample_rate: 16000.0
biquad: -1.9285085 0.9299964 -1.8999636 0.9499818 0.9499818
biquad: -1.9688775 0.9703966 -2.0000000 1.0000000 1.0000000
```

As you can see, a digital filter can be specified by $(5 \cdot N + 1)$ numbers, where N is the number of second-order sections, and 1 is the sample rate.

User Downloadable Filters can also be created by specialized third-party software such as MATLAB, available from The Mathworks, Inc. See Appendix B, "MATLAB Functions" (Page) for four MATLAB functions useful in working with downloadable filters and Audio Precision waveform files.

Audio Precision has also included with APWIN version 2.11 and later a DSP filter creation program from Momentum Data Systems called Filter Design Package for Audio Precision (FDP). See the separate *Filter Design Package User's Manual* ('Filter_Design_Package.pdf' installed in \Apwin\Documentation\UserManuals\) for instruction in the use of this program.

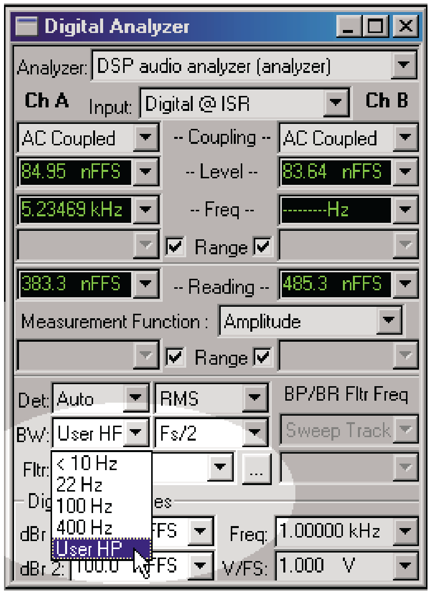
Using User Downloadable Filters in APWIN

You can choose a User Downloadable filter for the Digital Analyzer in the same way that you might choose one of the provided DSP filters.

With the Digital Analyzer display set to its large form (double-click on the panel Title Bar) the low-pass and high-pass filters are selected in the two fields to the right of the **BW** (bandwidth) designation. The weighting filters are selected in the **Fltr** (filter) field.

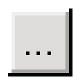
*The DSP filters are not available in every measurement function of the Digital Analyzer. Set **Measurement Function** to **Amplitude**, **2-Channel Ratio** or one of the two **THD+N** modes to choose DSP filters.*

Figure 13. User Downloadable Filter Choices



Click the arrow to drop down the list of filters for any of these fields. The last filter option on all lists selects the User Downloadable Filter appropriate for that setting: **User HP**, **User LP** or **User Weighting Filter.**

Although you can save many different user filter files for use in APWIN, you can only select one user file for any of the three filter positions at any one time. When APWIN is launched, by default no user filter files are selected. To choose your filters, click the ellipsis box to the right of the **Fltr** field. A file browser window will appear that will allow you to select filter files for all three the filter types: low-pass, high-pass and weighting. You can also view the filter **Info** string for any of the three selected filters. The default folder for user filter files is C:\Apwin\S2Cascade\Dspfilters.



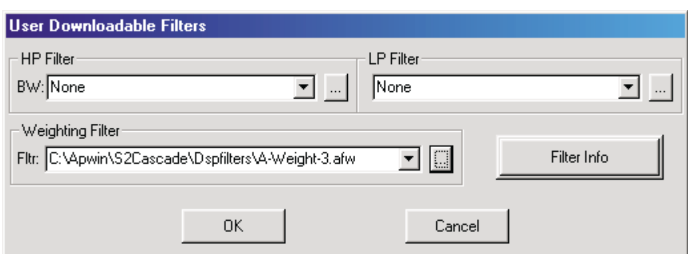


Figure 12. User Downloadable Filter File Browsers.

When you have chosen your user filter files and closed the browser window, notice that the **BW** and **Fltr** fields have been automatically set to

User to reflect your choices. As long as you do not exit APWIN or load a test file, these user filter files will remain attached to their respective filter lists. You still have the option, however, of choosing other APWIN-provided filters (or **None**) from the lists without losing the link to the file you have selected.

A test saved with an attached user filter will re-attach the filter to the Digital Analyzer when the test is loaded.

If you select **User** from any of the three filter lists and a user filter file has not been previously selected for that setting, the user filter file browser will appear and prompt you to choose a filter file.

Quasi-Anechoic Acoustical Tester (MLS)

Two new features have been added to the Quasi-Anechoic Acoustical Tester, also called the Maximum Length Sequence (MLS) tester. These additions have resulted in a rearrangement of the settings fields at the bottom of the Digital Analyzer panel when MLS is selected.

V 2.11

MLS Averaging

When measuring a coherent signal in the presence of uncorrelated noise, synchronous averaging of many measurements will reduce the noise reading and allow the coherent signal to be recovered more effectively. MLS averaging is done synchronously in the time domain. To enable MLS averaging, click the arrow by the new **Averages** field and select the number of readings to be averaged from the list.

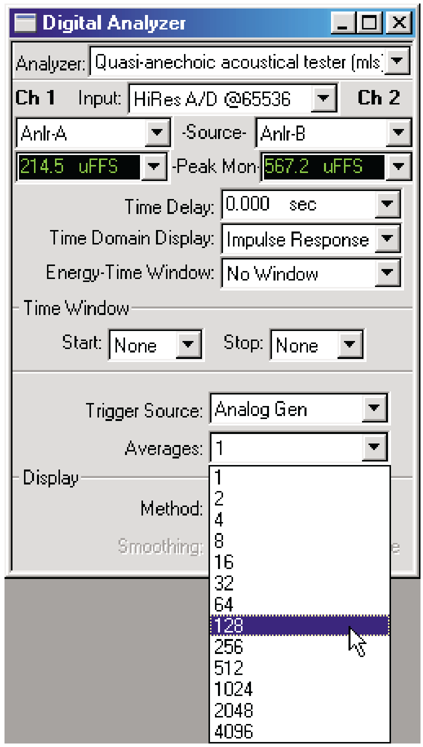


Figure 15. MLS Averaging choices.

MLS Octave Smoothing

V 2.11

Unlike MLS Averaging, which takes the average of a number of measurements, MLS Octave Smoothing is a display option that shows the results of one measurement as modified by a smoothing algorithm.

In previous versions of APWIN, the MLS display options **Interpolate**, **Display Samples** and **Peak Values** were available in the **Wave Display** field. These same display choices are still available in the **Method** field in the new Display area of the panel. You can select Octave Smoothing by clicking the new fourth option, **Smooth**.

Figure 14. Octave Smoothing Option.

When **Smooth** is chosen the **Smoothing** setting field becomes available. Here you can specify the degree of smoothing by entering values from 0 to 2.64 octaves.

Octave smoothing is a common technique in loudspeaker response measurement, useful in revealing trends by smoothing out anomalies in the response curve. The APWIN implementation uses a hybrid FFT bin averaging and interpolation technique to achieve smooth results even at very low bin densities. Smoothing, which only affects frequency-domain displays, effectively passes the raw response data through multiple constant-Q bandpass filters, one filter centered on each frequency requested from the Sweep panel. The bandwidth of these filters, in octaves, is specified in the **Smoothing** field.

V 2.11

Default Test Settings for Jitter Settling

Audio Precision provides many sample test files with APWIN, as well as a default test file which defines the base parameters for each new test you begin. The initial conditions for all these tests depend on the default settings saved in each test file.

As of APWIN version 2.11, the default test file settings for interface jitter settling have been made more discriminating to offer better performance. The new settings will more consistently recognize settled jitter amplitude readings during sweeps.

*Figure 17. **Interface Jitter** settings on the DIO section of the Settling panel. Shown above are the defaults used before APWIN version 2.11.*

Figure 16. Jitter settling, new defaults.

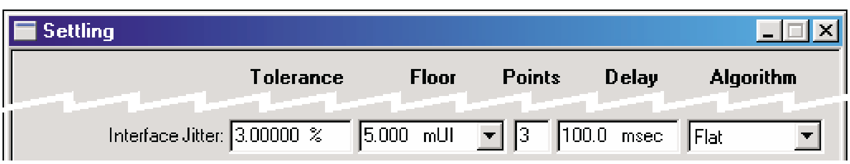


Figure 18. Jitter settling, new defaults in UI.

Using the previous defaults, the occurrence of several closely-spaced jitter amplitude readings in the sweep between desired data points would sometimes satisfy **Settling**; the last value of the series would then be reported as a settled reading. The Settling panel **Floor** default setting has been 3 ns, which is a rather broad range for many jitter amplitude readings. In the test files provided with APWIN version 2.11 and later, the **Floor** setting has been reduced to 500 ps; additionally, **Algorithm** has been set to **Flat**. Figures 17, 16 and 18 compare the old defaults to the new.

*If you choose to view jitter in UI rather than in seconds, the default **Floor** setting is 5 mUI, which is equivalent to about 800 ps at the default sampling rate of 48 kHz. However, if you should set a different sample rate, the value of the **Floor** setting in seconds will change, and with it the discrimination of settled jitter readings.*

This change is only a default setting for new tests. Older versions of tests will retain the settings incorporated in them.

Of course, you may change and save new settings at any time.

Analog Analyzer: AES17 Low-Pass Filter **V 2.11**

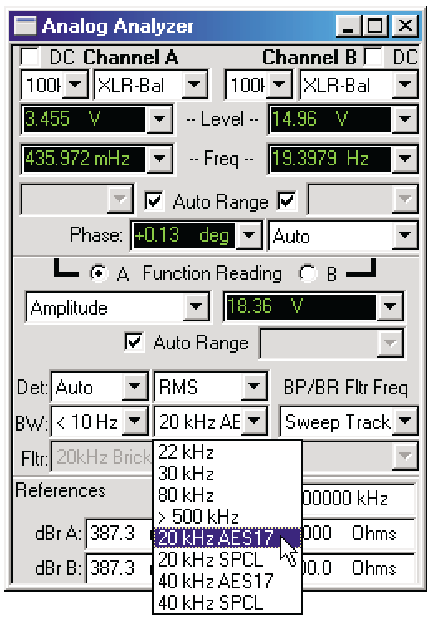


Figure 19. AES17 Low-Pass Filter Selection and Channel A routing.

AES17 specifies a standard low-pass filter for THD+N measurements of digital-to-analog converters (DACs) which exhibit high-level out-of-band noise.

Audio Precision now manufactures a new hardware filter for optional installation in System Two, System Two Cascade or System Two Cascade *Plus* which satisfies the AES17 specification. APWIN 2.2 and later versions support the use of this filter.

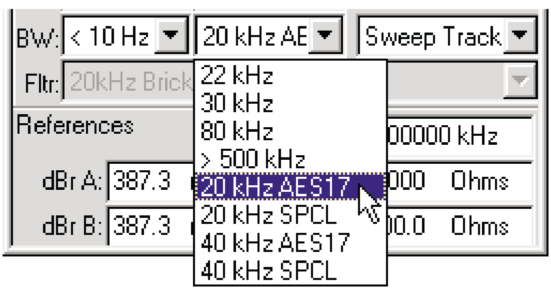
This new filter is designated the S-AES17 Low-Pass Filter Option, and replaces the earlier S2-AES17LP filter released in March of 2000. The new S-AES17 option offers improved performance and features compared to the earlier S2-AES17LP.

Contact your Audio Precision representative for information about ordering the S-AES17 option. The option consists of a dual-frequency pre-analyzer filter module, the SLPX; two additional analyzer option filters, FLP-B20K and FLP-B40K, and installation software and instructions.

An essential feature of the S-AES17 filter is its location *previous* to the Analog Analyzer, where it attenuates the out-of-band noise components before they can overload the circuitry and make distortion and other low-level measurements difficult. The additional option filters complete the job to satisfy the AES17 recommendation.

When installed, the AES17 filter is enabled by choosing one of the four new choices available on the low-pass **BW** (bandwidth) filter list.

*The four AES17 filter selections will not appear on the **BW** list until the filter installation software (which is provided with the filter kit) is run.*



The new selections are:

- **20 kHz AES17**
  This choice selects both the 20 kHz pre-analyzer filter and the 20 kHz brick-wall option filter. Selection of other option filters is disabled. If the required option filter is not installed, a warning will appear and the selection will default to **20 kHz SPCL**.

- **20 kHz SPCL**
  This choice selects only the 20 kHz pre-analyzer filter. You may choose any option filter, or **None**.
- **40 kHz AES17**
  This choice selects both the 40 kHz pre-analyzer filter and the 40 kHz brick-wall option filter. Selection of other option filters is disabled. If the required option filter is not installed, a warning will appear and the selection will default to **40 kHz SPCL**.
- **40 kHz SPCL**
  This choice selects only the 40 kHz pre-analyzer filter. You may choose any option filter, or **None**.

The pre-analyzer section of the filter operates on only one channel at a time and follows the **A** or **B** channel selection made when setting **Function Reading**, as shown in Figures 19 and 20.

In the 20 kHz AES17 mode, the combination of pre-analyzer filter and the option filter is designed to provide a flat passband (±0.1 dB) through 20 kHz with a stopband attenuation of 60 dB or better above 24 kHz, satisfying the AES17-1998 Section 4.2.1 specification for a “standard low-pass filter.”

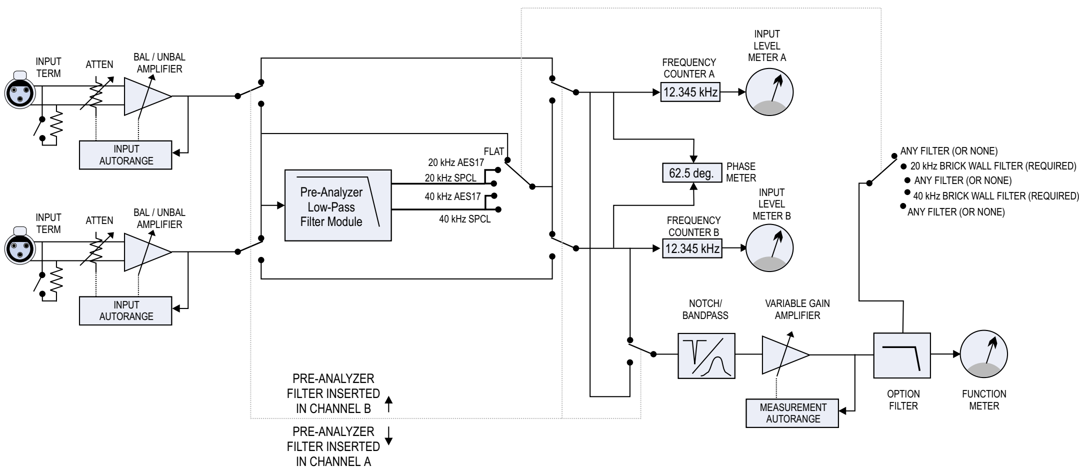


Figure 20. Block Diagram of AES17 Low-Pass Filter implementation in System Two, System Two Cascade or System Two Cascade Plus. *Note the location of the pre-analyzer section of the filter, and the linked impletmentation of the post-analyzer option filters.*

In the 40 kHz AES17 mode, the combination of pre-analyzer filter and the option filter is designed to provide a flat passband (±0.1 dB) through 40 kHz with a stopband attenuation of 60 dB or better above 48 kHz.

In either case, this filter will attenuate out-of-band noise typically created by oversampled converters and provide meaningful THD+N measurements of audio signals accompanied by high out-of-band noise. For detailed information see Appendix A, "AES17 Filter" (Page 89) and the documentation provided with the filter.

V 2.11

Digital I/O Panel: Rate Ref

The **Rate Ref** entry field on the Digital I/O panel will now accept a wider range of entry settings. The previous range was 8 kHz–192 kHz; in APWIN version 2.11 and later versions, any frequency value between 6.75 kHz and 216 kHz can be entered as the **Rate Ref**.

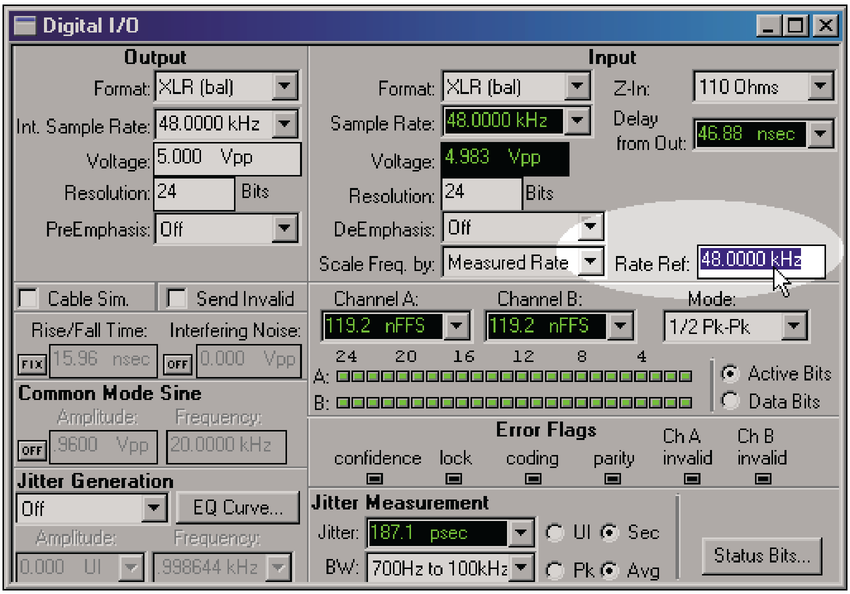


*Figure 21. Digital I/O Panel **Rate Ref** setting, set at 48.0000 kHz (System Two shown; System Two Cascade has similar appearance).*

The **Rate Ref** entry field has two uses, one interface-related and one digital audio-related. For digital audio measurements, this field serves as an absolutely stable value for the nominal sample rate when **DIO Rate Ref** is selected in the **Scale Freq. by** field. For serial digital interface parameter measurements, the **Rate Ref** value is the reference for all relative frequency units selectable in the **Sample Rate** (System Two Cascade: **Sample Rate-ISR**) display field. Thus, if it is desired to display measured sample rate in terms of PPM deviation from the nominal rate, enter the nominal rate into the **Rate Ref** field and select PPM units for the **Sample Rate** (System Two Cascade: **Sample Rate-ISR**) display.

*For System Two, **Scale Freq. by** is a digital input setting only, and when **Rate Ref** is selected in this field it serves as a reference for the input sample rate. For System Two Cascade under APWIN 2.11 and later, **Scale Freq. by** is also a digital output setting, and **Rate Ref** can be used as a reference for either (or both) input or output sample rate scaling. See **Digital Output Sample Rate Scaling** (Page 9).*

For more information about using **Rate Ref** settings, see Page 7-5 in the *APWIN User's Manual for System Two Version 2* or Page 7-6 in the *APWIN User's Manual for System Two Cascade Version 2.*

Shaped Burst Performance Improved V 2.11

Previous limitations to the Digital Generator **Shaped Burst** duration have been raised from 2^{16} samples (65,536) to $(2^{23} - 1)$ samples (8,388,607) in APWIN version 2.14. This allows much longer shaped bursts to be specified.

Generator Auto ON/OFF Option V2.11

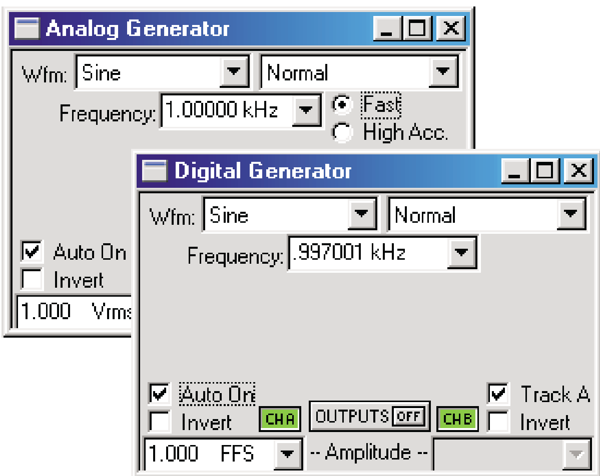


*Figure 22. **Auto On** checkboxes on Analog and Digital Generator panels.*

A new feature for all systems is the generator **Auto On** function. For both the Analog and Digital Generators, **Auto On** automatically switches the generator **ON** when a sweep starts, and **OFF** when the sweep terminates.

This feature will be of particular interest for those involved in power amplifier or loudspeaker testing. With the generator set to **OFF** and **Auto On** enabled, signal will only be applied to the DUT while the sweep is actually running.

The initial transient created when a generator is turned on requires a delay before measurement to allow the generator, DUT, ranging and analyzer circuits to stabilize. When **Auto On** is enabled, the **ON** command is applied to the generator at the moment the sweep starts. To avoid the effects of the switching transient, set the **Pre-Sweep Delay** on the expanded Sweep panel to a sufficient time, in many cases **.5 s** to **1 s** or more.

*Tests saved with the generator set to **ON** will open under the same condition, with the generator **ON**. **Auto On** only affects the generator state during a sweep.*

V 2.11 APWIN Support for New PCI Interface Card

Audio Precision has made the APIB interface available on both ISA and PCMCIA cards for a number of years. As a third option, you can now connect your PC to System One, System Two or System Two Cascade using a PCI APIB interface card. Software drivers for the PCI interface were included as part of APWIN version 2.11.

Your PC operating system must be either Microsoft® Windows 98®, Windows 2000®, Windows ME® or Windows NT® 4.0 to be compatible with the PCI interface. S1.EXE running under DOS or APWIN running under Windows 95® are not compatible with the PCI interface.

The APWIN PCI interface is recommended for use with System One analog domain (SYS-22a) and with all configurations of System Two and System Two Cascade.

To install a PCI-WIN interface card, first install the APWIN software on your computer. Then shut down the computer, disconnect its power, remove the cover and install the card in an unused PCI slot. Install the hold-down screw on the card bracket and replace the cover.

When the computer is turned on, the software should detect the PCI-WIN card and install the proper drivers.

For more information about installing APIB cards, see *APWIN Installation & Getting Started* for your System and APWIN version.

With version 2.14, the ISA, PCI and PCMCIA interface drivers have been further updated to improve performance and to broaden the combinations of interface cards and operating systems available.

See the latest APIB interface driver compatibility chart at audioprecision.com/techsupport/compatibility.htm.

Interface Drivers Folder

V 2.11

When APWIN is installed prior to the hardware installation of an ISA, PCI or PCMCIA APIB interface card, the drivers are normally installed by the Windows Plug and Play feature. If Windows prompts you to locate a driver manually, go to the Drivers folder (new in APWIN version 2.14), found in the APWIN CD-ROM root folder.

Changes to External Sweeps Operation

V 2.11

Background: Internal and External sweeps

Many of the sweeps performed in APWIN involve one of the internal generators, with the sweep following controlled parameters such as frequency or amplitude. The sweep and the generator share the control settings, and there is no ambiguity about sweep direction, or sweep **Start** or **Stop** values, for example.

However, it is often necessary to have APWIN track a sweep source that is not under internal control, such as a CD alignment disc, a recorder alignment tape, or a remote sweep generator. This is accomplished in the external sweep mode. When performing an external sweep, APWIN monitors the input signal and extracts sweep controls from the signal itself.

This information is always less certain than the mutually shared settings of an internal sweep, and APWIN requires some hints: the range and direction of the sweep, under what conditions to graph a point, when to end the sweep, and so on. With real-world signals, satisfying these criteria can be difficult, and external sweeps can sometimes fail to start or stop, or gather too many or too few points, or graph spurious points. As of APWIN version 2.11, several changes have been made to improve the performance of external sweeps.

*If you have performed external sweeps with earlier versions of APWIN, read **“Start On” Rules** (Page 28) carefully to determine which external sweep rule sets the behavior you prefer. The default rule is **Within Start Tolerance,** which corresponds to the external sweep behavior in APWIN versions 2.01 through 2.10. The Start On rule can be changed with an OLE command, as explained on Pages 28 and 29.*

New External Sweep Operation

External sweeps are driven by actual measurements. The **Source 1** field on the Sweep Panel must be configured as a meter reading rather than a generator setting, and the reading must satisfy the requirements entered in the Settling Panel for that meter. For a frequency sweep, the meter will be one of the frequency meters. For amplitude-controlled sweeps a level or amplitude meter is normally used.

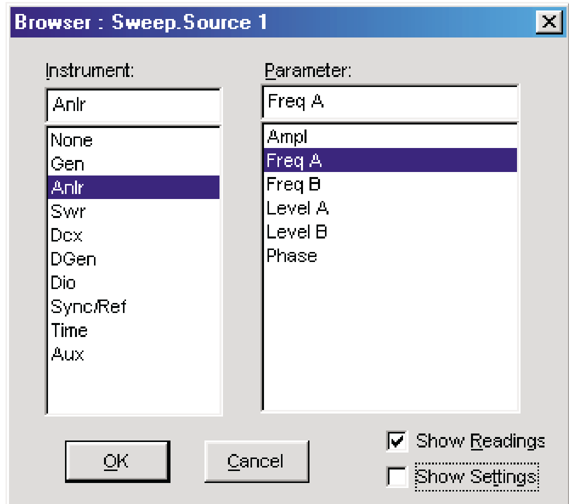


Figure 23. Sweep Source 1 Browser, set to Freq A meter readings.

*Other meters may be selected for special or difficult measurements. For example, the DSP analyzer in **Bandpass** mode can be used to control an amplitude sweep with Analog Analyzer measurements. The narrow bandpass filter in the DSP analyzer allows measurements with much-reduced sensitivity to noise.*

Start and Stop

The extent and direction of the sweep are set by the **Start** and **Stop** fields. If the value of **Start** is less than **Stop**, the sweep will proceed upward, in the direction of lesser to greater. If **Start** is greater than **Stop**, the sweep will proceed downward.

Spacing

The **Spacing** field helps to determine the step size within the sweep. **Spacing** is entered as a percentage, and the range for each new data point is the value of the last point plotted plus or minus the **Spacing** percentage. At the beginning of a sweep, the Start On rules (below) determine whether or not **Spacing** affects the acquisition of the first plotted point.

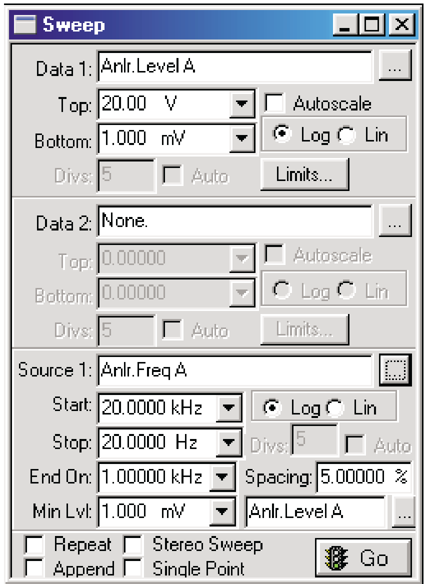


Figure 24. Sweep Panel in External Sweep Mode.

End On

The **End On** field in the Source 1 section of the Sweep panel allows you to set the conditions under which an External Sweep test will terminate. Of course, an External Sweep test can be manually terminated by pressing the **Esc** key or clicking on the **Stop** icon when it is apparent that the sequence of test tones has ended, but **End On** (or the equivalent OLE command, `AP.Sweep.Source1.EndOn`, if in a procedure) can terminate the sweep automatically.

End On now follows this logic:

- if the value set in **End On** equals the **Stop** value (plus or minus the **Spacing** percentage), the sweep will terminate when the **End On** value occurs;
- if the value set in **End On** does *not* equal **Stop** (±**Spacing**) and is outside the range set by **Start** and **Stop**, the sweep will terminate when the **End On** value occurs;
- if the value set in **End On** does not equal **Stop** (±**Spacing**) and is *within* the range set by **Start** and **Stop**, the sweep will terminate on a second occurrence of the **End On** value.

This produces several different behaviors:

- The first case, where **End On** equals **Stop** (±**Spacing**), allows you to set **End On** at the end of your desired sweep. The last value acquired (**Stop ±Spacing**) will be graphed.

- The second case, where **End On** does not equal **Stop** (±**Spacing**) and is outside the **Start–Stop** range, allows you to set **End On** beyond your desired sweep. You can, for example, place an out-of-band cue tone after the swept tones to terminate the sweep.
- The third case, where **End On** does not equal **Stop** (±**Spacing**) and is within the **Start–Stop** range, allows you to set **End On** to a value in the mid-range of your desired sweep. This accommodates, for example, a frequency test which sweeps across the desired spectrum and then returns to a mid-spectrum tone. If **End On** is set to that mid-spectrum tone, it will note the first occurrence of that tone as the sweep passes through it, and at the return the second occurrence of the tone will terminate the sweep. Many pre-recorded test tapes and CDs are made just this way, with, for example, a 20 Hz to 20 kHz sweep followed by a 1 kHz reference tone. When performing split-site measurements with an Audio Precision System One or System Two, the remote generator can be set up so that the generator dwells at its Generator panel setting (which can be a mid-band reference frequency) before and after a Source 1 frequency sweep.

Min Lvl

The value set in the **Min Lvl** (minimum level) field allows you to exclude any measurements below a preset level, as measured by the meter specified in the field just to the right of **Min Lvl**. This acts as a "noise gate," preventing noise or low-level signals from interfering with the process. When **Min Lvl** is set very low, all signals will exceed the setting and will be plotted; when **Min Lvl** is set very high, no signals will exceed the setting and nothing will be plotted.

"Start On" Rules

An APWIN sweep is initiated by pressing **F9** or clicking **GO**. In an internal sweep, two things happen in response to this: the generator you have selected begins its sweep, and the acquisition of sweep measurements is enabled. In an external sweep, of course, APWIN does not start a generator, but **F9** (or **GO**) enables the acquisition of sweep measurements in the same way. However, no data will be acquired until the externally-generated signal satisfies the conditions set by the Start On rule which is in effect. There are three Start On rules. The default is

- **Within Start Tolerance.** (OLE command control 0).
  The sweep starts on a settled reading that is above the **Min Lvl** value, and that is also within the tolerance set by applying the **Spacing** percentage to the **Start** value. For example, if **Start** is 20 Hz and **Spacing** is 5 %, a reading between 19 and 21 Hz will

start the sweep. Or, if **Start** is 1 V rms and **Spacing** is 10 %, an amplitude between 0.9 and 1.1 V rms will start the sweep. This rule was new with APWIN version 2.01, and is the default behavior for APWIN version 2.11 and later versions.

This rule governs the normal behavior of external sweeps. By using OLE commands (a feature which was new with APWIN version 2.11), the advanced user can set one of two other Start On rules.

- **Beyond Start Value.** (OLE command control 1).
  The sweep starts on a settled reading that is above the **Min Lvl** value and exceeds the **Start** value in the sweep direction (up or down, as determined by the relative values of the **Start** and **End** settings). No data is collected until the **Source1** meter reading passes (or equals) the **Start** value. This rule was new with APWIN version 2.11.
- **Any Settled Reading**. (OLE command control 2).
  The sweep starts on a settled reading that is above the **Min Lvl** value, without regard to the value in the **Start** field. Sweep measurements are collected in the direction set by the **Start** and **Stop** field values. In a sweep with **Start** at 20 Hz and **Stop** at 20 kHz, a 1 kHz measurement will not accept the next value until 1 kHz +**Spacing** is satisfied. Under this rule, the data collected can include values above or below the sweep graph limits set in **Start** and **Stop**. This rule was the default behavior in APWIN version 2.00 and earlier.

External Sweep OLE Commands

If you would like to read other meters in APWIN while the sweep is running, use:

```
AP.Sweep.StartNoWait
```

To select one of the three Start On rules listed above, use:

```
AP.Sweep.External.StartOnRule
```

If you are sitting at the keyboard, you can manually start and stop the sweep. However, if the external sweep is within a procedure, it may hang the program while you are away from your desk. These commands can be used to escape a running sweep:

```
AP.Sweep.AbortTime
AP.Application.SetWatchDogTimer1
AP.Application.SetWatchDogTimer2
```

Some Hints

In a glide sweep, the signal is gliding continuously from one value to another and will never settle as it will in a step sweep. In an external frequency glide sweep it is essential that the frequency **Settling Algorithm** be set to **None**, the **Points** to **1**, and **Delay** to **0**.

Also, check your **Min Lvl** settings. Acting rather like a noise gate, **Min Lvl** sets an amplitude threshold below which data will not be acquired; this is useful in directing the sweep to ignore the silence between tracks on a test CD, for example. The default meter assigned to make the **Min Lvl** reading is **Anlr A**, but a browser can offer you a selection of sources. Be sure the meter that you have selected is the appropriate meter for your test, that you have considered the effect of any filters you may have attached to the meter, and that the **Min Lvl** threshold setting is at the correct level.

V 2.11

Installation Program Warning

The APWIN Installation Program now detects previous installations of APWIN and displays a warning message. You are then given the choice to continue, to choose a different folder for installation, or to abort the installation altogether.

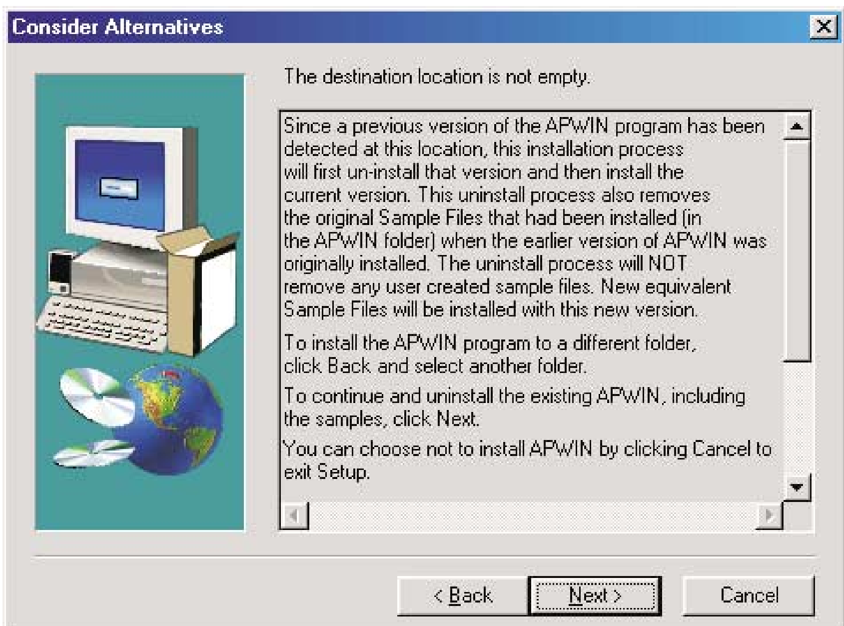


Figure 25. Installation warning

See *APWIN Installation & Getting Started* for your platform and version for detailed information about installing APWIN.

Japanese Language O.S. Support V 2.11

Conflicts in character code usage caused APWIN to operate improperly under the Japanese language version of Microsoft Windows 2000. This has been fixed in APWIN version 2.14.

New Audio Track List V 2.11

The APWIN version 2.14 CD-ROM audio test recording tracks have undergone some modifications and deletions, with the result that the CD track numbering has changed. Check the track list in the CD-ROM booklet.

New Sample Files V 2.11

With APWIN 2.14, New sample files for External Sweeps ("X-files") have been added. Check the file list in the CD-ROM booklet.

Chapter 3: AP Basic

AP Basic Extensions

This chapter features both OLE commands which have changed and new OLE commands which have been added to APWIN Basic Extensions to provide programmatic control for new features.

AP.Anlr.FuncFilterId — Property

Syntax **`AP.Anlr.FuncFilterId`**

Data Type Integer — This command controls the selection of the Analog Analyzer Function Meter Filter.
ID numbers are used to select the appropriate filter.
Refer to Analog Filter ID List in Appendix E of the APWIN Basic Extensions Manual to obtain filter identification numbers.

Description This command selects one (or none) of the available weighting filters. The weighting filters are optional filters that plug internally into the analyzer. An attempt to select a filter that is not present will result in the filter being set to NONE.

See Also `AP.Anlr.FuncFilter`, `AP.Anlr.FuncFilterHP`, `AP.Anlr.FuncFilterLP`

Example

```
Sub Main
  AP.Application.NewTest    'Reset panels
  AP.Gen.Output = True
  AP.Anlr.ChAInput = 2
```

```
  AP.Sweep.Data1.Id = 5906  'Set Sweep Data 1 to
     "Anlr.Ampl"
  AP.Anlr.FuncFilterId = 12017 '"A" Weighting filter.
  Debug.Print AP.Anlr.FuncFilterId
End Sub
```

Example Output

```
12017
```

AP.Compute.Avg.StartUnit

Method

Syntax **AP.Compute.Avg.StartUnit**

Result String

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

See Also AP.Compute.Avg.StopUnit

Example

```
Sub Main
  AP.Compute.Avg.Data1 = True
  AP.Compute.Avg.Start("Hz") = 0.02
  Debug.Print AP.Compute.Avg.StartUnit
End Sub
```

Output Hz

AP.Compute.Avg.StopUnit

Method

Syntax **AP.Compute.Avg.StopUnit**

Result String

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

See Also AP.Compute.Avg.StartUnit

Example

```
Sub Main
  AP.Compute.Avg.Data1 = True
  AP.Compute.Avg.Stop("Hz") = 20000.0
```

```
  Debug.Print AP.Compute.Avg.StopUnit
End Sub
```

Output Hz

AP.Compute.Center.StartUnit

Method

Syntax **AP.Compute.Center.StartUnit**

Result String

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

See Also AP.Compute.Center.StopUnit

Example

```
Sub Main
  AP.Compute.Center.Data1 = True
  AP.Compute.Center.Start("Hz") = 20000.0
  Debug.Print AP.Compute.Center.StartUnit
End Sub
```

Output Hz

AP.Compute.Center.StopUnit

Method

Syntax **AP.Compute.Center.StopUnit**

Result String

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

See Also AP.Compute.Center.StartUnit

Example

```
Sub Main
  AP.Compute.Center.Data1 = True
  AP.Compute.Center.Stop("Hz") = 20000.0
  Debug.Print AP.Compute.Center.StopUnit
End Sub
```

Output Hz

AP.Compute.Invert.HorizontalUnit Method

Syntax **`AP.Compute.Invert.HorizontalUnit`**

Result String

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

See Also `AP.Compute.Invert.Horizontal`

Example

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

Output Hz

AP.Compute.Linearity.StartUnit Method

Syntax **`AP.Compute.Linearity.StartUnit`**

Result String

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

See Also `AP.Compute.Linearity.StopUnit`

Example

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

Output Hz

AP.Compute.Linearity.StopUnit

Method

Syntax **`AP.Compute.Linearity.StopUnit`**

Result String

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

See Also `AP.Compute.Center.StartUnit`

Example

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

Output Hz

AP.Compute.Max.StartUnit

Method

Syntax **`AP.Compute.Max.StartUnit`**

Result String

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

See Also `AP.Compute.Max.StopUnit`

Example

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

Output Hz

AP.Compute.Max.StopUnit

Method

Syntax **`AP.Compute.Max.StopUnit`**

Result String

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

See Also `AP.Compute.Max.StartUnit`

Example

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

Output Hz

AP.Compute.Min.StartUnit

Method

Syntax **`AP.Compute.Min.StartUnit`**

Result String

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

See Also `AP.Compute.Min.StopUnit`

Example

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

Output Hz

AP.Compute.Min.StopUnit — Method

Syntax `AP.Compute.Min.StopUnit`

Result String

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

See Also `AP.Compute.Min.StartUnit`

Example

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

Output Hz

AP.Compute.Normalize.HorizontalUnit — Method

Syntax `AP.Compute.Normalize.HorizontalUnit`

Result String

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

See Also `AP.Compute.Normalize.Horizontal`

Example

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

Output Hz

AP.Compute.Normalize.TargetUnit — Method

Syntax **`AP.Compute.Normalize.TargetUnit`**

Result String

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

See Also `AP.Compute.Normalize.Target`

Example

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

Output V

AP.Compute.Sigma.StartUnit — Method

Syntax **`AP.Compute.Sigma.StartUnit`**

Result String

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

See Also `AP.Compute.Sigma.Start`, `AP.Compute.Sigma.StopUnit`

Example

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

Output Hz

AP.Compute.Sigma.StopUnit

Method

Syntax **`AP.Compute.Sigma.StopUnit`**

Result String

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

See Also `AP.Compute.Sigma.StartUnit`

Example

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

Output Hz

AP.Compute.Status.Id

Method

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

Data Type

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

Parameter

Name	Description
Num	Number representing computation order.

Description This command returns the Compute Status Identification Number.

See Also `AP.Compute.Status.NumOf`

Example

```
Sub Main
  AP.File.OpenTest("Status Id.at2c")
  Computations = AP.Compute.Status.NumOf
  Debug.Print  Computations & " Computations
performed in the following order."
  For Counter = 0 To Computations - 1
     Debug.Print
ComputationText(AP.Compute.Status.Id(Counter))
  Next Counter
End Sub
Function ComputationText(IdNum)
  Select Case(IdNum)
     Case 142
        ComputationText = "Normalize"
     Case 138
        ComputationText = "Invert"
     Case 144
        ComputationText = "Smooth"
     Case 139
        ComputationText = "Linearity"
     Case 136
        ComputationText = "Center"
     Case 137
        ComputationText = "Delta"
     Case 135
        ComputationText = "Average"
     Case 140
        ComputationText = "Minimum"
     Case 141
        ComputationText = "Maximum"
     Case 151
        ComputationText = "Equalize"
  End Select
End Function
```

Output

10 Computations performed in the following order.

Normalize
Invert
Smooth
Linearity

Center
Delta
Average
Minimum
Maximum
Equalize

AP.Compute.Status.NumOf — Method

Syntax **`AP.Compute.Status.NumOf`**

Result Integer

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

See Also `AP.Compute.Status.Id`

Example See example for `AP.Compute.Status.Id`.

AP.Data.Status — Property

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

Data Type Boolean

`True`	When setting: set *`Status`* to `True`. When reading: *`Status`* is `True` (active).
`False`	When setting: set *`Status`* to `False`. When reading: *`Status`* is `False` (inactive).

Parameter

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 Data Column (0-7).

Index	This value defines the 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.
Status	`apbInvalid` = Data displayed as Invalid. `apbTimeout` = Data displayed as Timed out. `apbUnregulated` = Data displayed as Unregulated.

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

Example

```
Sub Main
  Dim Timeouts As Integer
  Timeouts = 0
  AP.File.OpenTest("Timeouts.at2c")
  For Row = 0 To (AP.Data.ColSize(0, 1) - 1) Step 1
    Timeout = AP.Data.Status(0, 1, Row, apbTimeout)
    If Timeout = True Then Timeouts = Timeouts + 1
  Next Row
  Debug.Print Timeouts & " timeouts detected."
End Sub
```

Output 2 timeouts detected.

AP.DGen.EqCurveColumn

Get Only Property

Syntax **`AP.DGen.EqCurveColumn(`**`ByVal` *`Data`* `As Integer`**`)`**

Data Type Integer Column number.

Parameter

Name	Description
Data	Number of the Sweep Data (1-6) of the data in memory.

Description This command returns the column number in the attached file used in the Digital Generator EqCurve waveform selection.

See Also `AP.Sweep.Data.AutoDiv, AP.Sweep.Data.LogLin`

AP.DGen.EqCurveFilename

Get Only Property

Syntax **`AP.DGen.EqCurveFilename`**

Data Type Integer Any valid DOS filename and extension.

Description This command returns the File Name of the attached file used for the Digital Generator EqCurve waveform selection.

See Also `AP.DGen.EqCurve, AP.DGen.EqCurveColumn`

AP.Gen.EqCurveColumn

Get Only Property

Syntax **`AP.Gen.EqCurveColumn(`**`ByVal` *`Data`* `As Integer`**`)`**

Data Type Integer Column number

Parameter

Name	Description
Data	Number of the Sweep Data (1-6) of the data in memory.

Description This command returns the column number in the attached file used in the Analog Generator EqCurve waveform selection.

See Also `AP.Sweep.Data.AutoDiv, AP.Sweep.Data.LogLin, AP.Gen.EqCurve, AP.Gen.EqCurveFilename`

AP.Gen.EqCurveFilename

Get Only Property

Syntax **`AP.Gen.EqCurveFilename`**

Data Type Integer Any valid DOS filename and extension.

Description This command returns the File Name of the attached file used for the Analog Generator EqCurve waveform selection.

See Also `AP.Gen.EqCurve, AP.Gen.EqCurveColumn`

AP.Graph.AddComment

Method

Syntax `AP.Graph.AddComment(ByVal Text As String)`

Data Type Void

Parameter

Name	Description
Text	ASCII text.

Description This command appends the ASCII characters to the comment section in the Graph panel.

See Also `AP.Graph.Comment`, `AP.Graph.CommentShow`

AP.Graph.CompanyNameShow

Property

Syntax `AP.Graph.CompanyNameShow`

Data Type Boolean

True	Display Company Name in the graph window title bar.
False	Remove Company Name from the graph window title bar.

Description This command displays or removes the company name from the title bar on the graph window.

See Also `AP.Graph.CommentShow`

Example See example for `AP.Graph.Comment`.

AP.Graph.Label

Property

Syntax `AP.Graph.Label(ByVal AxisId As Constant)`

Data Type String ASCII text.

Parameter

Name	Description
AxisId	apbAxisTop = Top center. apbAxisBottom = Bottom center. apbAxisLeft = Left center. apbAxisRight = Right center.

Description This command set or returns the graph axis Labels.

Example

```
Sub Main
   AP.Application.NewTest
   AP.Gen.OutputOn = True
   AP.Anlr.ChAInput = 1
   AP.Sweep.Start
   AP.Graph.Title = "Title and Labels Example"
   AP.Graph.LabelAuto(apbAxisLeft) = False
   AP.Graph.Label(apbAxisLeft) = "Left"
   AP.Graph.LabelAuto(apbAxisBottom) = False
   AP.Graph.Label(apbAxisBottom) = "Bottom"
   AP.Graph.LabelAuto(apbAxisRight) = False
   AP.Graph.Label(apbAxisRight) = "Right"
   AP.Graph.LabelAuto(apbAxisTop) = False
   AP.Graph.Label(apbAxisTop) = "Top"
End Sub
```

AP.Graph.LabelAuto

Property

Syntax **AP.Graph.LabelAuto(**ByVal *AxisId* As Constant**)**

Data Type Boolean

True	Label text generated automatically based on the Sweep Panel settings.
False	Label defined programmatically.

Parameter

Name	Description
AxisId	apbAxisTop = Top center. apbAxisBottom = Bottom center. apbAxisLeft = Left center. apbAxisRight = Right center.

Description This command specifies whether the graph labels are automatically generated based on the Sweep Panel settings or programmatically.

Example See example for `AP.Graph.Label`.

AP.Graph.Legend.Comment

Property

Syntax **AP.Graph.Legend.Comment(**ByVal *SweepId* As Integer, ByVal *TraceId* As Integer**)**

Data Type String ASCII characters

Parameter

Name	Description
SweepId	Sweep number
TraceId	Trace number

Description This command transfers the ASCII characters between the Legend Trace Comment section in the Graph panel and a string variable.

Example

```
Sub Main
  AP.Application.NewTest
  AP.AGen.OutputOn = True
  AP.AnalogIn.Source(apbChA) = apbAnalogInGenMon
  AP.Sweep.Start
  AP.Graph.Legend.LineColor(1, 1) = apbRed
  AP.Graph.TraceShow(1, 1) = True
  AP.Graph.Legend.LineThickness(1, 1) = 1
  AP.Graph.Legend.LineStyle(1, 1) = apbSolid
  AP.Graph.Legend.Comment(1, 1) = "Trace Comment."
End Sub
```

AP.Graph.Legend.LineColor

Property

Syntax **AP.Graph.Legend.LineColor(**ByVal *SweepId* As Integer, ByVal *TraceId* As Integer**)**

Data Type Constant

apbBlue
apbCyan
apbGray
apbGreen
apbMagenta
apbRed
apbYellow

Parameter

Name	Description
SweepId	Sweep number
TraceId	Trace number

Description This command sets the Legend Trace Color for the specified Sweep Trace.

Example See example for `AP.Graph.Legend.Comment`.

AP.Graph.Legend.LineStyle Property

Syntax **`AP.Graph.Legend.LineStyle(`**`ByVal` *`SweepId`* `As Integer, ByVal` *`TraceId`* `As Integer`**`)`**

Data Type Constant

apbDash
apbDashDot
apbDashDotDot
apbDot
apbSolid

Parameter

Name	Description
SweepId	Sweep number
TraceId	Trace number

Description This command sets the Legend Trace Line Style for the specified Sweep Trace.

Example See example for `AP.Graph.Legend.Comment.`

AP.Graph.Legend.LineThickness Property

Syntax **`AP.Graph.Legend.LineThickness(`**`ByVal` *`SweepId`* `As Integer, ByVal` *`TraceId`* `As Integer`**`)`**

Data Type Integer 1-32

Parameter

Name	Description
SweepId	Sweep number.
TraceId	Trace number.

Description This command sets the Line Thickness for the specified Sweep Trace.

Example See example for `AP.Graph.Legend.Comment`.

AP.Graph.RefDataClear

Method

Syntax **`AP.Graph.RefDataClear`**

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

```
Sub Main
  AP.Graph.RefDataClear
  AP.Application.NewTest
  AP.AGen.OutputOn = True
  AP.AnalogIn.Source(apbChA) = apbAnalogInGenMon
  AP.Sweep.Start
  AP.Graph.Data(1).LogLin = apbLin
  AP.Graph.RefDataStore
  AP.Graph.RefDataShow = True
  AP.Graph.OptimizeLeft
  AP.Graph.CopyToSweepPanel
  AP.Sweep.Start
End Sub
```

AP.Graph.RefDataShow

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 visible or invisible on the graph.

See Also `AP.Graph.RefDataClear, AP.Graph.RefDataStore`

Example See example for `AP.Graph.RefDataClear`.

AP.Graph.RefDataStore — Method

Syntax **`AP.Graph.RefDataStore`**

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`.

AP.Graph.ScrollBarsOn — Property

Syntax **`AP.Graph.ScrollBarsOn`**

Data Type Boolean

True	Display Scroll Bars.
False	Remove Scroll Bars from view.

Description This command makes the Scroll Bars visible or invisible on the graph.

AP.Graph.Sweeps — Method

Syntax **`AP.Graph.Sweeps`**

Data Type Integer

Description This command returns the number of Sweeps contained in the current data set.

See Also `AP.Graph.SweepTraces`

AP.Graph.SweepShow

Property

Syntax `AP Graph.SweepShow(ByVal SweepId As Integer)`

Data Type Boolean

True	Display Sweep Data.
False	Remove Sweep Data from view.

Parameter

Name	Description
SweepId	Sweep Data number.

Description This command makes the specified Sweep set of traces visible or invisible on the graph.

See Also `AP.Graph.Sweeps`

AP.Graph.SweepsTrace

Method

Syntax `AP.Graph.SweepsTrace`

Data Type Integer

Description This command returns the number of Sweeps contained in the current data set.

See Also `AP.Graph.Sweeps`

AP.Graph.TimeDateShow

Property

Syntax `AP.Graph.TimeDateShow`

Data Type Boolean

True	Display Time and Date.
False	Remove Time and Date from view.

Description This command displays or removes from view the Time and Date in the title bar section in the Graph panel.

See Also `AP.Graph.Title`

AP.Graph.Title

Property

Syntax **`AP.Graph.Title`**

Data Type String ASCII characters.

Description This command transfers the ASCII characters between the Title section in the Graph panel and a string variable.

See Also `AP.Graph.TimeDateShow`

AP.Graph.TraceShow

Property

Syntax **`AP.Graph.TraceShow(`**`ByVal` *`SweepId`* `As Integer, ByVal` *`TraceId`* `As Integer`**`)`**

Data Type Constant

apbBlue
apbCyan
apbGray
apbGreen
apbMagenta
apbRed
apbYellow

Parameter

Name	Description
SweepId	Sweep number
TraceId	Trace number

Description This command makes the specified Trace visible or invisible on the graph.

See Also `AP.Graph.Sweeps, AP.Graph.SweepTraces`

AP.Graph.ZoomOriginal

Method

Syntax `AP.Graph.ZoomOriginal`

Data Type Void

Description This command replaces the current zoomed-in view with the graph coordinates in use when the most recent sweep was started, or with the default initial graph coordinates if no sweep has yet been made since APWIN was launched.

See Also `AP.Graph.ZoomOut`

AP.Graph.ZoomOut

Method

Syntax `AP.Graph.ZoomOut`

Data Type Void

Description This command causes the most recent zoom view to be replaced with the previous one. If you have zoomed repeatedly, the coordinates of each zoom have been saved in sequence in a stack. You may then Zoomout repeatedly to work back up through the stack, viewing the series of zoomed views in reverse order.

See Also `AP.Graph.ZoomOriginal`

AP.PSIA.MasterClkDir

Property

Syntax `AP.PSIA.MasterClkDir`

Data Type

Integer	Transmit-side master clock	Receive-side master clock
0	Input	Input
1	Output	Input
2	Input	Output

Description This command selects the master clock direction for transmit and receive sides simultaneously. Each master clock port can be configured as an input or as an output, although not all combinations are available. See the table above. In input (slave) mode, the master clock is provided by an external source. In output (master) mode, the master clock is provided by the PSIA.

See Also

`AP.PSIA.Rx.MasterClk.Factor, AP.PSIA.Rx.BitClkDir, AP.PSIA.Rx.FrameClkDir`

Example

```
Sub Main
  AP.S2CDio.OutFormat = 3    ' PSIA output
  AP.PSIA.MasterClkDir = 1   ' Tx out, Rx in
  AP.PSIA.OutputsOn = True   ' Outputs on
  AP.PSIA.VoltageSetting = PSIA_3_3_TTL
                             ' 3.3 V TTL
  AP.PSIA.Tx.MasterClk.Factor = 256
                             ' master clk = 256 * Fs
  AP.PSIA.Tx.NFsClk.Factor = 128
                             ' N*Fs clk = 128 * Fs
  AP.PSIA.Tx.NFsClk.InvWfm = False
                             ' non-inverted
  AP.PSIA.Rx.MasterClk.Factor = 128
                             ' N*Fs clk = 128 * Fs
  AP.PSIA.Rx.NFsClk.Factor = 128
                             ' master clk = 128 * Fs
  AP.PSIA.Rx.NFsClk.InvWfm = True
                             ' inverted
End Sub
```

AP.PSIA.OutputsOn — Property

Syntax **`AP.PSIA.OutputsOn`**

Data Type Boolean

True	On
False	Off

Description This command turns the PSIA outputs on or off. When the outputs are off, they are tri-stated. When the outputs are on, they are driven according to the voltage setting.

See Also `AP.PSIA.VoltageSetting`

Example See `AP.PSIA.MasterClkDir`.

AP.PSIA.Rx.BitClk.Dir
AP.PSIA.Tx.BitClk.Dir

Property

Syntax **`AP.PSIA.Rx.BitClk.Dir`**

`AP.PSIA.Tx.BitClk.Dir`

Data Type Integer

0	Output
1	Input

Description This command selects the bit clock direction. Each bit clock port can be configured as an output or as an input. In output (master) mode, the bit clock is provided by the PSIA. In input (slave) mode, the bit clock is provided by an external source.

See Also `AP.PSIA.Rx.BitClk.Factor`, `AP.PSIA.Rx.FrameClk.Dir`, `AP.PSIA.Rx.MasterClkDir`

Example

```
Sub Main
  AP.PSIA.Tx.BitClk.Dir = 0       ' output
  AP.PSIA.Tx.BitClk.Factor = 32   ' 32-bit words
  AP.PSIA.Rx.BitClk.Dir = 1       ' input
  AP.PSIA.Rx.BitClk.Factor = 32   ' 32-bit words
End Sub
```

AP.PSIA.Rx.BitClk.Factor
AP.PSIA.Tx.BitClk.Factor

Property

Syntax **`AP.PSIA.Rx.BitClk.Factor`**

`AP.PSIA.Tx.BitClk.Factor`

Data Type Integer 8-32 (limited also by digital resolution settings)

Description This command specifies the ratio (factor) between the bit clock and the channel clock. It is equal to the number of bits per channel. It cannot be set lower than the number of bits specified in the digital output resolution field (for Tx) or the digital input resolution field (for Rx). The maximum number of bits per channel is 32.

See Also `AP.PSIA.Rx.BitClk.Dir`, `AP.S2CDio.InResolution`, `AP.S2CDio.OutResolution`

Example See `AP.PSIA.Rx.BitClk.Dir`.

AP.PSIA.Rx.ChannelClk.BitWidePulse AP.PSIA.Tx.ChannelClk.BitWidePulse

Property

Syntax

`AP.PSIA.Rx.ChannelClk.BitWidePulse`

`AP.PSIA.Tx.ChannelClk.BitWidePulse`

Data Type Boolean

True	Bit Wide Pulse (one period of the bit clock)
False	Approximately 50% duty cycle

Description This command selects the pulse width of the channel clock output. Assuming that the channel clock output is not inverted, the following are true:

- When ChannelClk.BitWidePulse is *True*, the channel clock is high for the first bit of each subframe, and low for the rest of the subframe.
- When ChannelClk.BitWidePulse is *False*, and the number of bits B is even, the channel clock is high for the first B/2 bits, and low for the rest of the subframe.
- When Channel.BitWidePulse is *False*, and the number of bits B is odd, the channel clock is high for the first (B-1)/2 bits, and low for the rest of the subframe.

See Also `AP.PSIA.Rx.ChannelClk.EdgeSync`, `AP.PSIA.Rx.ChannelClk.Factor`, `AP.PSIA.Rx.ChannelClk.InvWfm`

Example

```
Sub Main
  AP.PSIA.Tx.ChannelClk.BitWidePulse = False
                           ' 50% duty cycle
```

```
    AP.PSIA.Tx.ChannelClk.EdgeSync = 0
                         ' assert on rising edge
    AP.PSIA.Tx.ChannelClk.Factor = 2
                         ' 2 channels
    AP.PSIA.Tx.ChannelClk.InvWfm = True
                         ' invert channelclk
    AP.PSIA.Tx.ChannelClk.BitWidePulse = False
                         ' 50% duty cycle
    AP.PSIA.Rx.ChannelClk.EdgeSync = 1
                         ' latch on falling edge
    AP.PSIA.Rx.ChannelClk.Factor = 2
                         ' 2 channels
    AP.PSIA.Rx.ChannelClk.InvWfm = False
                         ' inverted channelclk
End Sub
```

AP.PSIA.Rx.ChannelClk.EdgeSync
AP.PSIA.Tx.ChannelClk.EdgeSync

Property

Syntax **`AP.PSIA.Rx.ChannelClk.EdgeSync`**

`AP.PSIA.Tx.ChannelClk.EdgeSync`

Data Type Integer

0	Rising edge
1	Falling edge

Description For the transmitter side (Tx), this command selects whether the channel clock output is asserted at the rising or falling edge of the bit clock. For the receiver side (Rx), this command selects whether the channel clock input is latched at the rising or falling edge of the bit clock.

See Also `AP.PSIA.Rx.ChannelClk.BitWidePulse`, `AP.PSIA.Rx.ChannelClk.Dir`, `AP.PSIA.Rx.ChannelClk.InvWfm`.

Example See `AP.PSIA.Rx.ChannelClk.BitWidePulse`.

AP.PSIA.Rx.ChannelClk.Factor
AP.PSIA.Tx.ChannelClk.Factor

Property

Syntax **`AP.PSIA.Rx.ChannelClk.Factor`**

`AP.PSIA.Tx.ChannelClk.Factor`

Data Type Long 1–256

Description This command specifies the ratio (factor) between the channel clock and the frame clock. It is equal to the number of channels per frame. The minimum number of channels is 1. The maximum number of channels is 256; limitations on the master clock rate may further restrict this.

See Also `AP.PSIA.Rx.ChannelClk.BitWidePulse`, `AP.PSIA.Rx.ChannelClk.EdgeSync`, `AP.PSIA.Rx.ChannelClk.InvWfm`.

Example See `AP.PSIA.Rx.ChannelClk.BitWidePulse`, `AP.PSIA.Rx.ChannelClk.InvWfm`.

AP.PSIA.Rx.ChannelClk.InvWfm
AP.PSIA.Tx.ChannelClk.InvWfm

Property

Syntax **`AP.PSIA.Rx.ChannelClk.InvWfm`**
`AP.PSIA.Tx.ChannelClk.InvWfm`

Data Type Boolean

True	Inverted channel clock
False	Non-inverted channel clock

Description This command sets the polarity of the channel clock. When set to *False* (non-inverted), the channel clock is high at the start of the subframe, and low for the rest of the subframe. When set to *True* (inverted), the channel clock is low at the start of the subframe, and high for the rest of the subframe.

See Also `AP.PSIA.Rx.ChannelClk.BitWidePulse`, `AP.PSIA.Rx.ChannelClk.EdgeSync`, `AP.PSIA.Rx.ChannelClk.Factor`

Example See `AP.PSIA.Rx.ChannelClk.BitWidePulse`.

AP.PSIA.Rx.Data.ChannelA
AP.PSIA.Tx.Data.ChannelA

Property

Syntax **`AP.PSIA.Rx.Data.ChannelA`**

`AP.PSIA.Tx.Data.ChannelA`

Data Type Integer — 0 to the one less than the number of channels specified by the associated ChannelClk.Factor command

Description For the transmitter side (Tx), this command causes generator Channel A data to appear on the selected subframe. For the receiver side (Rx), this command causes data from the selected subframe to be applied to Channel A of the analyzer.

Note that the channel assignments are zero-based, that is, the channels are numbered from zero to one less than the number of available channels.

See Also `AP.PSIA.Rx.Data.ChannelB`

Example See `AP.PSIA.Rx.Data.EdgeSync`.

AP.PSIA.Rx.Data.ChannelB
AP.PSIA.Tx.Data.ChannelB

Property

Syntax **`AP.PSIA.Rx.Data.ChannelB`**

`AP.PSIA.Tx.Data.ChannelB`

Data Type Integer — 0 to the one less than the number of channels specified by the associated ChannelClk.Factor command

Description For the transmitter side (Tx), this command causes generator Channel B data to appear on the selected subframe. For the receiver side (Rx), this command causes data from the selected subframe to be applied to Channel B of the analyzer.

Note that the channel assignments are zero-based, that is, the channels are numbered from zero to one less than the number of available channels.

See Also `AP.PSIA.Rx.Data.ChannelA`

Example See `AP.PSIA.Rx.Data.EdgeSync`.

AP.PSIA.Rx.Data.EdgeSync
AP.PSIA.Tx.Data.EdgeSync

Property

Syntax **`AP.PSIA.Rx.Data.EdgeSync`**

`AP.PSIA.Tx.Data.EdgeSync`

Data Type Integer

0	Rising edge
1	Falling edge

Description For the transmitter side (Tx), this command selects whether the data output is asserted at the rising or falling edge of the bit clock. For the receiver side (Rx), this command selects whether the data input is latched at the rising or falling edge of the bit clock.

See Also `AP.PSIA.Rx.ChannelClk.EdgeSync`, `AP.PSIA.Rx.FrameClk.EdgeSync`

Example

```
Sub Main
  AP.PSIA.Tx.ChannelClk.Factor = 4
               ' 4 channels...
  AP.PSIA.Tx.BitClk.Factor = 32
               ' ...of 32-bit data
  AP.PSIA.Tx.Data.EdgeSync = 0
               ' assert on rising edge
  AP.PSIA.Tx.Data.ChannelA = 1
               ' assign ChA data to channel 1
  AP.PSIA.Tx.Data.ChannelB = 3
               ' assign ChB data to channel 3
```

```
    AP.PSIA.Tx.Data.MsbFirst = True
                  ' send audio word MSB first
    AP.PSIA.Tx.Data.PrePadType = 2
                  ' pre-pad with sign
    AP.PSIA.Tx.Data.PostPadType = 0
                  ' post-pad with zeros
       ' Note: the following two lines are equivalent
    AP.PSIA.Tx.Data.Justify(apbRight)
                  ' right justify audio word
    AP.PSIA.Tx.Data.PadBits =
    AP.PSIA.Tx.BitClk.Factor -
    AP.S2CDio.OutResolution
    AP.PSIA.Rx.ChannelClk.Factor = 4
                  ' 4 channels...
    AP.PSIA.Rx.BitClk.Factor = 32
                  ' ...of 32-bit data
    AP.PSIA.Rx.Data.EdgeSync = 1
                  ' latch on falling edge
    AP.PSIA.Rx.Data.ChannelA = 1
                  ' channel 1 data -> ChA of analyzer
    AP.PSIA.Rx.Data.ChannelB = 3
                  ' channel 1 data -> ChB of analyzer
    AP.PSIA.Rx.Data.MsbFirst = True
               ' accept audio word MSB first
               ' Note: the following two lines are
equivalent
    AP.PSIA.Rx.Data.Justify(apbRight)
               ' accept right-justified audio word
    AP.PSIA.Rx.Data.PadBits =
    AP.PSIA.Rx.BitClk.Factor -
    AP.S2CDio.InResolution
End Sub
```

AP.PSIA.Rx.Data.Justify AP.PSIA.Tx.Data.Justify — Method

Syntax **AP.PSIA.Rx.Data.Justify(**ByVal *Justify* As Constant**)**

AP.PSIA.Tx.Data.Justify(ByVal *Justify* As Constant**)**

Parameter

Name	Description	
Justify	apbLeft:	Left justify audio word
	apbRight:	Right justify audio word

Description This command justifies the audio data to the first bit of the subframe (apbLeft) or the last bit of the subframe (apbRight). For left justification, any padding bits trail the audio word. For right justification, any padding bits lead the audio word. Note that justification does not affect the bit order in the word (that is, whether the MSB or the LSB comes first).

See Also AP.PSIA.Tx.Data.PostPadType, AP.PSIA.Tx.Data.PrePadType, AP.PSIA.Rx.Data.MsbFirst

Example See AP.PSIA.Rx.Data.EdgeSync.

AP.PSIA.Rx.Data.MSBFirst AP.PSIA.Tx.Data.MSBFirst

Property

Syntax **AP.PSIA.Rx.Data.MSBFirst**

AP.PSIA.Tx.Data.MSBFirst

Data Type Boolean

True	MSB first
False	LSB first

Description For the transmitter side (Tx), this command specifies whether audio data is sent Most Significant Bit (MSB) first or Least Significant Bit (LSB) first. For the receiver side (Rx), this command specifies whether audio data is accepted MSB first or LSB first.

See Also AP.PSIA.Rx.Data.Justify

Example See AP.PSIA.Rx.Data.EdgeSync.

AP.PSIA.Rx.Data.PadBits AP.PSIA.Tx.Data.PadBits

Property

Syntax **`AP.PSIA.Rx.Data.PadBits`**

`AP.PSIA.Tx.Data.PadBits`

Data Type Long — 0–24 (limited also by the number of bits per channel and the digital resolution)

Description For the transmitter side (Tx), this command sets the number of leading (leftmost) pad bits. If the sum of the number of pad bits and the number of bits in the audio word is less than the number of bits per channel, the subframe will also be padded with trailing bits. For the receiver side (Rx), this command sets the offset in bits of the audio data in the subframe, that is, the number of bits that will be skipped before audio data is clocked in.

See Also `AP.PSIA.Tx.Data.PostPadType`, `AP.PSIA.Tx.Data.PrePadType`, `AP.PSIA.Rx.BitClk.Factor`, `AP.S2CDio.InResolution`, `AP.S2CDio.OutResolution`

Example See `AP.PSIA.Rx.Data.EdgeSync`.

AP.PSIA.Rx.FrameClk.BitWidePulse AP.PSIA.Tx.FrameClk.BitWidePulse

Property

Syntax **`AP.PSIA.Rx.FrameClk.BitWidePulse`**

`AP.PSIA.Tx.FrameClk.BitWidePulse`

Data Type Boolean

`True`	Bit-wide pulse (one period of the bit clock)
`False`	Approximately 50% duty cycle

Description This command selects the pulse width of the frame clock output. Assuming that the frame clock output is not inverted, and not set to shift 1 bit left, the following are true:

- When FrameClk.BitWidePulse is *`True`*, the frame clock is high for the first bit of each frame, and low for the rest of the frame.

- When FrameClk.BitWidePulse is `False`, and the number of channels C is *even*, the frame clock is high for the first C/2 subframes, and low for the rest of the frame.
- When FrameClk.BitWidePulse is `False`, and the number of channels C is odd, the frame clock is high for the first (C-1)/2 subframes, and low for the rest of the frame.

Note: this command is not available when the associated frame clock direction is set to IN.

See Also `AP.PSIA.Rx.FrameClk.Dir`, `AP.PSIA.Rx.FrameClk.Rate`, `AP.PSIA.Rx.FrameClk.EdgeSync`, `AP.PSIA.Rx.FrameClk.InvWfm`, `AP.PSIA.Rx.FrameClk.ShiftOneBitLeft`

Example

```
Sub Main
AP.PSIA.Tx.FrameClk.Dir = 0
                ' output
  AP.PSIA.Tx.FrameClk.EdgeSync = 0
                ' assert on bitclk rise
  AP.PSIA.Tx.FrameClk.InvWfm = True
                ' invert
  AP.PSIA.Tx.FrameClk.ShiftOneBitLeft = True
                ' shift one bit left
  AP.PSIA.Tx.FrameClk.BitWidePulse = False
                ' 50% duty cycle
  AP.PSIA.Tx.FrameClk.Rate("Hz") = 44100
                ' CD sample rate
  AP.PSIA.Rx.FrameClk.Dir = 1
                ' input
  AP.PSIA.Rx.FrameClk.EdgeSync = 1
                ' latch on bitclk fall
  AP.PSIA.Rx.FrameClk.InvWfm = True
                ' inverted
  AP.PSIA.Rx.FrameClk.ShiftOneBitLeft  = True
                ' shifted one bit left
  AP.PSIA.Rx.FrameClk.Rate("Hz") = 44100
                ' CD sample rate
End Sub
```

AP.PSIA.Rx.FrameClk.Dir
AP.PSIA.Tx.FrameClk.Dir

Property

Syntax **`AP.PSIA.Rx.FrameClk.Dir`**

`AP.PSIA.Tx.FrameClk.Dir`

Data Type Integer

0	Output
1	Input

Description This command selects the frame clock direction. Each frame clock port can be configured as an output or as an input. In output (master) mode, the frame clock is provided by the PSIA. In input (slave) mode, the frame clock is provided by an external source.

See Also `AP.PSIA.Rx.FrameClk.BitWidePulse`, `AP.PSIA.Rx.FrameClk.EdgeSync`, `AP.PSIA.Rx.FrameClk.InvWfm`, `AP.PSIA.Rx.FrameClk.Rate`, `AP.PSIA.Rx.FrameClk.ShiftOneBitLeft`

Example See `AP.PSIA.Rx.FrameClk.BitWidePulse`.

AP.PSIA.Rx.FrameClk.EdgeSync
AP.PSIA.Tx.FrameClk.EdgeSync

Property

Syntax **`AP.PSIA.Rx.FrameClk.EdgeSync`**

`AP.PSIA.Tx.FrameClk.EdgeSync`

Data Type Integer

0	Rising edge
1	Falling edge

Description When the direction of the associated frame clock is set to OUT, this command selects whether the frame clock output is asserted at the rising or falling edge of the bit clock. When the direction of the associated frame clock is set to IN, this command selects whether the frame clock input is latched at the rising or falling edge of the bit clock.

See Also `AP.PSIA.Rx.FrameClk.BitWidePulse`, `AP.PSIA.Rx.FrameClk.Dir`, `AP.PSIA.Rx.FrameClk.InvWfm`, `AP.PSIA.Rx.FrameClk.Rate`, `AP.PSIA.Rx.FrameClk.ShiftOneBitLeft`

Example See `AP.PSIA.Rx.FrameClk.BitWidePulse`.

AP.PSIA.Rx.FrameClk.InvWfm
AP.PSIA.Tx.FrameClk.InvWfm

Property

Syntax **`AP.PSIA.Rx.FrameClk.InvWfm`**
`AP.PSIA.Tx.FrameClk.InvWfm`

Data Type Boolean

True	Inverted frame clock
False	Non-inverted frame clock

Description This command sets the polarity of the frame clock. When set to *False* (non-inverted), the frame clock is high at the start of the frame, and low for the rest of the frame. When set to *True* (inverted), the frame clock is low at the start of the frame, and high for the rest of the frame.

See Also `AP.PSIA.Rx.FrameClk.BitWidePulse`, `AP.PSIA.Rx.FrameClk.Dir`, `AP.PSIA.Rx.FrameClk.EdgeSync`, `AP.PSIA.Rx.FrameClk.Rate`, `AP.PSIA.Rx.FrameClk.ShiftOneBitLeft`

Example See `AP.PSIA.Rx.FrameClk.BitWidePulse`.

AP.PSIA.Rx.FrameClk.Rate
AP.PSIA.Tx.FrameClk.Rate

Property

Syntax **`AP.PSIA.Rx.FrameClk.Rate(`**`ByVal` *`Unit`* `As String`**`)`**
`AP.PSIA.Tx.FrameClk.Rate(``ByVal` *`Unit`* `As String`**`)`**

Data Type Double

Parameter

Name	Description
Unit	The following unit is available: Hz

Description When the direction of the associated frame clock is set to OUT, FrameClk.Rate sets the frequency of the frame clock output in Hz. Typically this is equal to the sample rate of the digital audio stream. When the direction of the associated frame clock is set to IN, FrameClk.Rate is used only to compute the displayed rates in the 'computed rate' column on the PSIA panels.

See Also `AP.PSIA.Rx.FrameClk.BitWidePulse`, `AP.PSIA.Rx.FrameClk.Dir`, `AP.PSIA.Rx.FrameClk.EdgeSync`, `AP.PSIA.Rx.FrameClk.InvWfm`, `AP.PSIA.Rx.FrameClk.ShiftOneBitLeft`

Example See `AP.PSIA.Rx.FrameClk.BitWidePulse`.

AP.PSIA.Rx.FrameClk.ShiftOneBitLeft AP.PSIA.Tx.FrameClk.ShiftOneBitLeft

Property

Syntax

`AP.PSIA.Rx.FrameClk.ShiftOneBitLeft`

`AP.PSIA.Tx.FrameClk.ShiftOneBitLeft`

Data Type Boolean

True	Frame clock valid one bit time before start of frame
False	Frame clock valid at start of frame

Description This command allows the frame clock to be asserted (when associated frame clock direction is OUT) or latched (when associated frame clock direction is IN) one bit time before the actual start of the frame. Typically, this is used in the I^2S bus standard. When FrameClk.ShiftOneBitLeft is *False*, the frame clock is asserted or latched at the start of the frame. When FrameClk.ShiftOneBitLeft is *True*, the frame clock is asserted or latched one bit time before the start of the frame.

See Also `AP.PSIA.Rx.FrameClk.BitWidePulse`, `AP.PSIA.Rx.FrameClk.Dir`,

`AP.PSIA.Rx.FrameClk.EdgeSync,`
`AP.PSIA.Rx.FrameClk.InvWfm,`
`AP.PSIA.Rx.FrameClk.Rate`

Example See `AP.PSIA.Rx.FrameClk.BitWidePulse`.

AP.PSIA.Rx.I2S
AP.PSIA.Tx.I2S

Method

Syntax **`AP.PSIA.Rx.I2S`**

`AP.PSIA.Tx.I2S`

Description This command configures the transmitter or receiver settings to be compatible with the Philips I^2S (Inter-IC Sound) bus.

See Also `AP.PSIA.Rx.FrameClk.ShiftOneBitLeft`

Example

```
Sub Main
     AP.PSIA.Tx.I2S ' I2S output format
     AP.PSIA.Tx.LoopBack ' copy settings to receiver
End Sub
```

AP.PSIA.Rx.MasterClk.Factor
AP.PSIA.Tx.MasterClk.Factor

Property

Syntax **`AP.PSIA.Rx.MasterClk.Factor`**

`AP.PSIA.Tx.MasterClk.Factor`

Data Type Long 1 or more

Description This command specifies the ratio (factor) between the master clock and the frame clock. Depending on other clock settings, certain factors may not be achievable.

Note: this command is not available when the associated master clock direction is set to OUT.

See Also `AP.PSIA.MasterClkDir`

Example See `AP.PSIA.MasterClkDir`, `AP.PSIA.Rx.NFsClk.Factor`.

AP.PSIA.Rx.NFsClk.Factor AP.PSIA.Tx.NFsClk.Factor

Property

Syntax **`AP.PSIA.Rx.NFsClk.Factor`**
`AP.PSIA.Tx.NFsClk.Factor`

Data Type Long 1 or more

Description This command specifies the ratio (factor) between the N*Fs clock and the frame clock. Depending on other clock settings, certain factors may not be achievable.

See Also `AP.PSIA.Rx.NFsClk.InvWfm`

Example See `AP.PSIA.MasterClkDir`.

AP.PSIA.Rx.NFsClk.InvWfm AP.PSIA.Tx.NFsClk.InvWfm

Property

Syntax **`AP.PSIA.Rx.NFsClk.InvWfm`**
`AP.PSIA.Tx.NFsClk.InvWfm`

Data Type Boolean

`True`	Inverted N*Fs clock
`False`	Non-inverted N*Fs clock

Description This command sets the polarity of the N*Fs clock. When set to *`False`* (non-inverted), the N*Fs clock is high at the start of the frame, and low for the rest of the frame. When set to *`True`* (inverted), the N*Fs clock is low at the start of the frame, and high for the rest of the frame.

See Also `AP.PSIA.Rx.NFsClk.Factor`

Example See `AP.PSIA.MasterClkDir`.

AP.PSIA.Tx.BitClk.Dir

Property

See `AP.PSIA.Rx.BitClk.Dir`

AP.PSIA.Tx.BitClk.Factor

Property

See `AP.PSIA.Rx.BitClk.Factor`

AP.PSIA.Tx.ChannelClk.BitWidePulse

Property

See `AP.PSIA.Rx.ChannelClk.BitWidePulse`

AP.PSIA.Tx.ChannelClk.EdgeSync

Property

See `AP.PSIA.Rx.ChannelClk.EdgeSync`

AP.PSIA.Tx.ChannelClk.Factor

Property

See `AP.PSIA.Rx.ChannelClk.Factor`

AP.PSIA.Tx.ChannelClk.InvWfm

Property

See `AP.PSIA.Rx.ChannelClk.InvWfm`

AP.PSIA.Tx.Data.ChannelA

Property

See `AP.PSIA.Rx.Data.ChannelA`

AP.PSIA.Tx.Data.ChannelB
Property

See `AP.PSIA.Rx.Data.ChannelB`

AP.PSIA.Tx.Data.EdgeSync
Property

See `AP.PSIA.Rx.Data.EdgeSync`

AP.PSIA.Tx.Data.Justify
Method

See `AP.PSIA.Rx.Data.Justify`

AP.PSIA.Tx.Data.MSBFirst
Property

See `AP.PSIA.Rx.Data.MSBFirst`

AP.PSIA.Tx.Data.PadBits
Property

See `AP.PSIA.Rx.Data.PadBits`

AP.PSIA.Tx.Data.PostPadType
Property

Syntax **`AP.PSIA.Tx.Data.PostPadType`**

Data Type Integer

0	Low: Set post (trailing) padding bits to logical low
1	High: Set post (trailing) padding bits to logical high
2	First bit: Set post (trailing) padding bits to the state of the last bit of the audio word

Description This command selects the value of the pad bits that trail the audio word. All pad bits have the same value: logical low, logical high, or the same state as the last bit in the audio word. In a two's

complement coding scheme, the MSB is the sign bit.
Therefore if the audio word is ordered LSB first, and `AP.PSIA.Tx.Data.PostPadType` = 2, then the audio word will be sign extended by the trailing pad bits.

See Also `AP.PSIA.Tx.Data.PrePadType`, `AP.PSIA.Rx.Data.PadBits`

Example See `AP.PSIA.Rx.Data.EdgeSync`.

AP.PSIA.Tx.Data.PrePadType

Property

Syntax **`AP.PSIA.Tx.Data.PrePadType`**

Data Type Integer

0	Low: Set pre (leading) padding bits to logical low
1	High: Set pre (leading) padding bits to logical high
2	First bit: Set pre (leading) padding bits to the state of the first bit of the audio word

Description This command selects the value of the pad bits that lead the audio word. All pad bits have the same value: logical low, logical high, or the same state as the first bit in the audio word. In a two's complement coding scheme, the MSB is the sign bit.
Therefore if the audio word is ordered MSB first, and `AP.PSIA.Tx.Data.PrePadType` = 2, then the audio word will be sign extended by the leading pad bits.

See Also `AP.PSIA.Tx.Data.PostPadType`, `AP.PSIA.Rx.Data.PadBits`

Example See `AP.PSIA.Rx.Data.EdgeSync`.

AP.PSIA.Tx.FrameClk.BitWidePulse

Property

See `AP.PSIA.Rx.FrameClk.BitWidePulse`

AP.PSIA.Tx.FrameClk.Dir — Property

See `AP.PSIA.Rx.FrameClk.Dir`

AP.PSIA.Tx.FrameClk.EdgeSync — Property

See `AP.PSIA.Rx.FrameClk.EdgeSync`

AP.PSIA.Tx.FrameClk.InvWfm — Property

See `AP.PSIA.Rx.FrameClk.InvWfm`

AP.PSIA.Tx.FrameClk.Rate — Property

See `AP.PSIA.Rx.FrameClk.Rate`

AP.PSIA.Tx.FrameClk.ShiftOneBitLeft — Property

See `AP.PSIA.Rx.FrameClk.ShiftOneBitLeft`

AP.PSIA.Tx.I2S — Method

See `AP.PSIA.Rx.I2S`

AP.PSIA.Tx.LoopBack — Method

Syntax **`AP.PSIA.Tx.LoopBack`**

Description This command configures the receiver according to the current transmitter settings, to provide a way to check data integrity through the PSIA. The following external connections are required to complete the loopback configuration (BNC-BNC cables are supplied for this purpose):

Transmitter bit clock → receiver bit clock

Transmitter frame clock → receiver frame clock

Transmitter data → receiver data

Example See `AP.PSIA.Rx.I2S`.

AP.PSIA.Tx.MasterClk.Factor Property

See `AP.PSIA.Rx.MasterClk.Factor`

AP.PSIA.Tx.NFsClk.Factor Property

See `AP.PSIA.Rx.NFsClk.Factor`

AP.PSIA.Tx.NFsClk.InvWfm Property

See `AP.PSIA.Rx.NFsClk.InvWfm`

AP.PSIA.VoltageSetting Property

Syntax **`AP.PSIA.VoltageSetting`**

Data Type Constant

PSIA_1_8_CMOS	1.8 V CMOS
PSIA_2_4_CMOS	2.4 V CMOS
PSIA_3_3_CMOS	3.3 V CMOS
PSIA_3_3_TTL	3.3 V TTL
PSIA_5_TTL	5.0 V TTL

Description This command sets the input and output voltages according to the logic family and voltage supplied.

Note: the outputs must be on for signal to appear at the PSIA outputs.

See Also `AP.PSIA.OutputsOn`

Example See `AP.PSIA.MasterClkDir.`

AP.S2CDio.InDecode

Property

Syntax **`AP.S2CDio.InDecode`**

Data Type Integer

0	No data expansion applied.
1	Apply µ-Law decoding to data signal.
2	Apply A-Law decoding to data signal.

Description This command selects the Digital Interface Receive Data Format Expansion for no expansion, μ-Law decoding or A-Law decoding.

See Also `AP.S2CDio.OutEncode`

Example

```
Sub Main
  AP.S2CDio.InDecode = 1   'µ-Law decode
End Sub
```

AP.S2CDIO.InScaleFreq

Property

Syntax **`AP.S2CDIO.InScaleFreq`**

Data Type Integer

0	**Output Rate:** the value in the **Sample Rate-OSR** field near the top of the Output section of the DIO panel.
1	**Meas Input Rate:** the measured value in the **Sample Rate-ISR** field.
2	**Status Bits A:** the value of sample frequency encoded into the received channel A status bits.

3	**DIO Rate Ref:** the value defined by the `AP.S2CDio.RefRate` command.

Description This command selects a source from which the digital audio sample rate is determined. The frequency of embedded digital audio signals must be normalized by a digital sample rate before display, whether it is displayed as a numeric frequency counter display or as a frequency component on an FFT graph.

See Also `AP.S2CDIO.OutScaleFreq, AP.S2CDIO.RefRate`

Example

```
Sub Main
  AP.S2CDIO.InScaleFreq = 1  'use measured input rate
End Sub
```

AP.S2CDio.OutEncode Property

Syntax **`AP.S2CDio.OutEncode`**

Data Type Integer

0	No data compression applied.
1	Apply µ-Law encoding to data signal.
2	Apply A-Law encoding to data signal.

Description This command selects the Digital Interface Transmit Data Format Compression for no compression, μ-Law encoding or A-Law encoding.

See Also `AP.S2CDio.InDecode`

Example

```
Sub Main
  AP.S2CDio.OutEncode = 2   'A-Law Encode
End Sub
```

AP.S2CDIO.OutScaleFreq **Property**

Syntax **`AP.S2CDIO.OutScaleFreq`**

Data Type Integer

0	**Output Rate:** the value in the **Sample Rate-OSR** field near the top of the Output section of the DIO panel.
1	**Meas Input Rate:** the measured value in the **Sample Rate-ISR** field.
2	**Meas Output Rate:** the measured value at the parallel output port when Output **Format** is set to **Parallel**; otherwise, the value in the **Sample Rate-OSR** field.
3	**DIO Rate Ref:** the value defined by the `AP.S2CDio.RefRate` command.

Description This command sets the source for scaling for the audio frequency embedded in the digital output signal.

See Also `AP.S2CDIO.InScaleFreq, AP.S2CDIO.RefRate`

Example

```
Sub Main
  AP.S2CDIO.OutScaleFreq = 1 'use measured input rate
End Sub
```

AP.S2CDsp.Analyzer.FuncFilterHPUserDefined **Method**

Syntax **`AP.S2CDsp.Analyzer.FuncFilterHPUserDefined`**`(ByVal` *`FileName`* `As String)`

Parameters

Name	Description
FileName	Long Path and File Names permitted up to 128 characters. The file must be an APWIN high-pass filter file (.afh). Enter “None” for the file name to remove the User-Defined Filter.

Result Boolean

True	File attachment successful.
False	File attachment failed.

Description This command attaches a User-Defined Filter for use with the DSP Audio Analyzer High Pass filter selection. When the filter is attached, the design is tested to determine that there are not more than 2 second-order sections used to create the filter and that the filter is stable. If either of these conditions is not met then this command returns False. To select the attached User-Defined Filter select “User Defined” with the `AP.S2CDsp.Analyzer.FuncFilterHP` command.

See Also `AP.S2CDsp.Analyzer.FuncFilterHP`, `AP.S2CDsp.Analyzer.FuncFilterLPUserDefined`, `AP.S2CDsp.Analyzer.FuncFilterWeightingUserDefined`

Example

```
Sub Main
  AP.S2CDsp.Analyzer.FuncFilterHPUserDefined _
  ("My_HP-1.afh")
End Sub
```

AP.S2CDsp.Analyzer.FuncFilterId Property

Syntax **`AP.S2CDsp.Analyzer.FuncFilterId`**

Data Type Integer

Description This command returns the FuncFilterId used in the Analyzer Function Meter Filter.

See Also `AP.S2CDsp.Analyzer.FuncFilter`

Example `AP.S2CDsp.Analyzer.FuncFilter`

AP.S2CDsp.Analyzer.FuncFilterLPUserDefined Method

Syntax **AP.S2CDsp.Analyzer.FuncFilterLPUserDefined(**ByVal *FileName* As String**)**

Parameters

Name	Description
FileName	Long Path and File Names permitted up to 128 characters. The file must be an APWIN low-pass filter file (.afl). Enter “None” for the file name to remove the User-Defined Filter.

Result Boolean

True	File attachment successful.
False	File attachment failed.

Description This command attaches a User-Defined Filter for use with the DSP Audio Analyzer Low Pass filter selection. When the filter is attached, the design is tested to determine that there are not more than 3 second-order sections used to create the filter and that the filter is stable. If either of these conditions is not met then this command returns False. To select the attached User-Defined Filter select “User Defined” with the AP.S2CDsp.Analyzer.FuncFilterLP command.

See Also AP.S2CDsp.Analyzer.FuncFilterLP, AP.S2CDsp.Analyzer.FuncFilterHPUserDefined, AP.S2CDsp.Analyzer.FuncFilterWeightingUserDefined

Example

```
Sub Main
  AP.S2CDsp.Analyzer.FuncFilterLPUserDefined _
("My_LP-1.afl")
End Sub
```

AP.S2CDsp.Analyzer.FuncFilterWeightingUserDefined Method

Syntax **AP.S2CDsp.Analyzer.FuncFilterWeightingUserDefined (**ByVal *FileName* As String**)**

Parameters	**Name**	**Description**
	FileName	Long Path and File Names permitted up to 128 characters. The file must be an APWIN weighting filter file (.afw). Enter "None" for the file name to remove the User-Defined Filter.

Result Boolean

True	File attachment successful.
False	File attachment failed.

Description This command attaches a User-Defined Filter for use with the DSP Audio Analyzer Weighting filter selection. When the filter is attached, the design is tested to determine that there are not more than 4 second-order sections used to create the filter and that the filter is stable. If either of these conditions is not met then this command returns False. To select the attached User-Defined Filter select "User Defined" with the `AP.S2CDsp.Analyzer.FuncFilter` command.

See Also `AP.S2CDsp.Analyzer.FuncFilter`, `AP.S2CDsp.Analyzer.FuncFilterHPUserDefined`, `AP.S2CDsp.Analyzer.FuncFilterLPUserDefined`

Example

```
Sub Main
 AP.S2CDsp.Analyzer.FuncFilterWeightingUser _
Defined("My WT-1.afw")
End Sub
```

AP.S2CDsp.Mls.Averages

Property

Syntax **`AP.S2CDsp.Mls.Averages`**

Data Type Integer

Integer	Number of Averages
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 number of acquisitions for the averaging function of the Quasi-Anechoic Acoustical Tester (MLS).

When measuring a coherent signal in the presence of uncorrelated noise, synchronous averaging of many measurements will reduce the noise reading and allow the coherent signal to be recovered more effectively. MLS averaging is done synchronously in the time domain.

Example

```
Sub Main
 AP.S2CDsp.Mls.Averages = 9    'set MLS Av. to 512
End Sub
```

AP.S2CDsp.Mls.Smoothing — Property

Syntax **`AP.S2CDsp.Mls.Smoothing`**

Data Type Double   Range of Values: 0 to 2.64 octaves

Description This command controls the width of the MLS Smoothing algorithm in octave units.

Octave smoothing is a common technique in loudspeaker response measurement, useful in revealing trends by smoothing out anomalies in the response curve. The APWIN implementation uses a hybrid FFT bin averaging and interpolation technique to achieve smooth results even at very low bin densities. Smoothing, which only affects frequency-domain displays, effectively passes the raw response data through multiple constant-Q bandpass filters, one filter centered on each frequency requested from the Sweep panel.

Example

```
Sub Main
 AP.S2CDsp.Mls.Smoothing = .3333
      'set MLS smoothing to 1/3 octave
End Sub
```

AP.S2DIO.InScaleFreq — Property

Syntax `AP.S2DIO.InScaleFreq`

Data Type Integer

0	**Output Rate:** the value in the **Int. Sample Rate** field near the top of the Output section of the DIO panel.
1	**Meas Input Rate:** the measured value in the **Sample Rate** field near the top of the Input section of the DIO panel.
2	**Status Bits A:** the value of sample frequency encoded into the received channel A status bits.
3	**DIO Rate Ref:** the value defined by the `AP.S2Dio.RefRate` command.

Description This command selects a source from which the digital audio sample rate is determined. The frequency of embedded digital audio signals must be normalized by a digital sample rate before display, whether it is displayed as a numeric frequency counter display or as a frequency component on an FFT graph.

See Also `AP.S2DIO.OutScaleFreq, AP.S2DIO.RefRate`

Example

```
Sub Main
 AP.S2DIO.InScaleFreq = 1   'use measured input rate
End Sub
```

AP.S2DIO.OutScaleFreq — Property

Syntax `AP.S2DIO.OutScaleFreq`

Data Type Integer

0	**Output Rate:** the value in the **Int. Sample Rate** field near the top of the Output section of the DIO panel.
1	**Meas Input Rate:** the measured value in the **Sample Rate** field near the top of the Input section of the DIO panel.

2	**DIO Rate Ref:** the value defined by the `AP.S2Dio.RefRate` command.

Description This command sets the source for scaling for the audio frequency embedded in the digital output signal.

See Also `AP.S2DIO.InScaleFreq, AP.S2DIO.RefRate`

Example

```
Sub Main
 AP.S2DIO.OutScaleFreq = 1  'use measured input rate
End Sub
```

AP.Sweep.Data*n*.LowerLimit.Column
AP.Sweep.Data*n*.UpperLimit.Column — Property

Syntax

```
AP.Sweep.Data1.LowerLimit.Column
AP.Sweep.Data1.UpperLimit.Column
AP.Sweep.Data2.LowerLimit.Column
AP.Sweep.Data2.UpperLimit.Column
AP.Sweep.Data3.LowerLimit.Column
AP.Sweep.Data3.UpperLimit.Column
AP.Sweep.Data4.LowerLimit.Column
AP.Sweep.Data4.UpperLimit.Column
AP.Sweep.Data5.LowerLimit.Column
AP.Sweep.Data5.UpperLimit.Column
AP.Sweep.Data6.LowerLimit.Column
AP.Sweep.Data6.UpperLimit.Column
```

Data Type Integer

Description This command returns the column number of the limit attached to the associated sweep data.

Example See example for `AP.Sweep.Data`*n*`.Limits`.

AP.Sweep.Data*n*.LowerLimit.Filename
AP.Sweep.Data*n*.UpperLimit.Filename

Property

Syntax

```
AP.Sweep.Data1.LowerLimit.Filename
AP.Sweep.Data1.UpperLimit.Filename
AP.Sweep.Data2.LowerLimit.Filename
AP.Sweep.Data2.UpperLimit.Filename
AP.Sweep.Data3.LowerLimit.Filename
AP.Sweep.Data3.UpperLimit.Filename
AP.Sweep.Data4.LowerLimit.Filename
AP.Sweep.Data4.UpperLimit.Filename
AP.Sweep.Data5.LowerLimit.Filename
AP.Sweep.Data5.UpperLimit.Filename
AP.Sweep.Data6.LowerLimit.Filename
AP.Sweep.Data6.UpperLimit.Filename
```

Data Type String

Description This command returns the filename of the limit attached to the associated sweep data.

Example See example for `AP.Sweep.Datan.Limits`.

AP.Sweep.External.StartOnRule

Property

Syntax `AP.Sweep.External.StartOnRule`

Data Type Long

0 — **Within Start Tolerance:** (Default) Data collection begins as soon as a settled reading is detected that is within the **Start** value plus or minus the **Spacing** tolerance as specified by the `AP.Sweep.Source1.Start` and `AP.Sweep.Source1.Spacing` commands.

Note: this sweep ***Start On*** *behavior was exhibited in APWIN versions 2.01 through 2.10.*

1 — **Beyond Start Value:** Data collection begins as soon as a settled reading is detected that is within the **Start** and **Stop** values as specified by the

`AP.Sweep.Source1.Start` and `AP.Sweep.Source1.Stop` commands. Additional readings are retained as long as the readings satisfy the sweep direction and **Spacing** requirement as specified by the `AP.Sweep.Source1.Spacing` command.

2 — **Any Settled Reading:** Data collection begins as soon as a settled reading is detected. Additional readings are retained as long as the readings satisfy the sweep direction and Spacing requirements as specified by the `AP.Sweep.Source1.Start`, `AP.Sweep.Source1.Stop`, and `AP.Sweep.Source1.Spacing` commands.

*Note: this sweep **Start On** behavior was exhibited in APWIN versions 1.0 through 2.0.*

Description This command selects the external sweep Start On rule.

See Also `AP.Sweep.Source1.Start`, `AP.Sweep.Source1.Stop`, `AP.Sweep.Source1.Spacing`

Example

```
Sub Main
 AP.Sweep.External.StartOnRule = 1
 'use Beyond Start Value rule
End Sub
```

AP.Sweep.Source1.SweepTable.Column Property

Syntax **`AP.Sweep.Source1.SweepTable.Column`**

Result Integer

Description This command returns the column number of the attached sweep table in Source 1.

See Also `AP.Sweep.Source1.Table`

AP.Sweep.Source1.SweepTable.Filename Property

Syntax **`AP.Sweep.Source1.SweepTable.Filename`**

Result String

Description This command returns the file name of the attached sweep table in Source 1.

See Also `AP.Sweep.Source1.Table`

Appendix A
AES17 Low-Pass Filter

The Audio Precision S-AES17 Low-Pass Filter Option provides a sharp roll-off beyond 20 kHz early in the measurement path to insure that measurements accurately reflect the true performance within the audio band. As an additional feature, the S-AES17 filter has a switchable mode which offers a broader passband, setting the sharp roll-off beyond 40 kHz.

This filter is a hardware option which meets the requirements of AES17-1998 and can be added to System Two, System Two Cascade or System Two Cascade *Plus*. It is unlike other optional filters available from Audio Precision in that it uses a larger module, and also in that this module is inserted in a different location in the measurement circuit, just after the input conditioning amplifier and prior to any measurement circuits. The S-AES17 filter option also contains two option filters, FLP-B20K and FLP-B40K, which must be used in conjunction with the pre-analyzer low-pass module for proper operation.

The Audio Precision S-AES17 Low-Pass Filter Option improves upon the capabilities of the earlier S2-AES17LP filter, which is no longer available.

Introduction

Digital-to-analog converters frequently use delta-sigma modulation and oversampling techniques to achieve high performance at an affordable cost. The noise inherent in this type of conversion is pushed up in frequency out of the audio band by means of high-order noise shaping. The resultant signal is quiet within the audio range but carries large amounts of out-of-band noise.

Measuring the audio performance of such converters can be a challenge for most contemporary wide-band audio distortion and noise analyzers. These instruments are designed to characterize classic analog audio devices, which typically exhibit a noise floor spread evenly over the full spectrum of the analyzer, usually diminishing with increased frequency. Out-of-band noise and interference are expected to be low in amplitude in relation to the signal, so band-limiting or noise-weighting filters, if needed, are inserted at the end of the measurement path, following several gain stages. When measuring classic analog audio devices, this approach yields accurate and repeatable THD+N results.

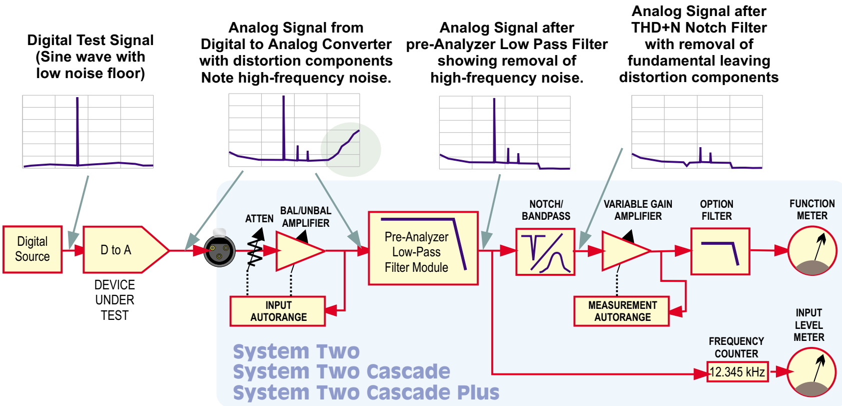


Figure 26. Conceptual signal path diagram showing how the AES17 Low-Pass Filter attenuates out-of-band noise and allows accurate measurement of audio-band distortion.

The spectrum of the noise floor of a noise-shaped delta-sigma D-to-A converter, however, shows a steeply rising energy characteristic beyond the 20 kHz upper limit of the audio band. When measuring low-level signals, the energy contribution of this ultrasonic noise can be substantial. In many situations it can overload instrument gain stages or throw off ranging circuits and cause grossly inaccurate measurements. Conventional band-limiting and noise-weighting filters cannot solve the problem because they are located too late in the measurement chain—the damage has already been done.

The AES17 Low-Pass Filter Specification

The Audio Engineering Society has defined the techniques for measuring digital audio equipment—that is, systems that contain digital-to-analog converters—in its standard, AES17. In section 4.2.1 AES17-1998 specifies the use of a “standard low-pass filter” that has a sharp roll-off above the audio band to attenuate out-of-band noise.

The AES17 standard specifies a stop-band attenuation of 60 dB or better above 24 kHz, quite a steep slope. The filter must be inserted early in the measurement path in order to remove the out-of-band noise before the measurement notch filter and its subsequent gain. This will insure that the "+N" part of THD+N contains only the in-band noise and distortion. Without the filter, the automatic gain ranging that normally follows the THD+N notch filter can behave incorrectly and the resulting measurement will be in error.

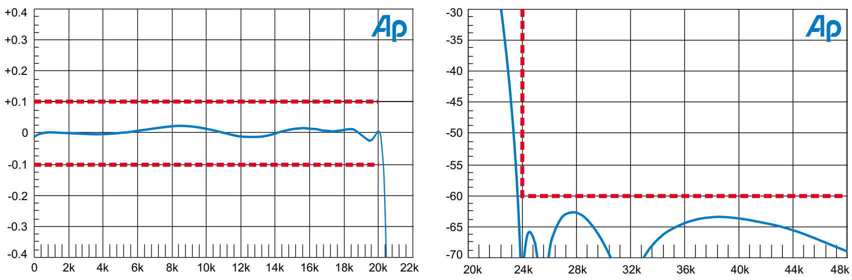


Figure 27. S-AES17 Low-Pass Filter Option passband and stopband response with 20 kHz setting, used with FLP-B20K option filter.

Figure 27 graphs the performance of the Audio Precision S-AES17 Low-Pass Filter Option at the **20 kHz** setting, used with the option filter FLP-B20K. This filter option satisfies the AES17 standard.

SPECIFICATIONS

The S-AES17 Low-Pass Filter Option is intended for measuring the THD+N of D-to-A converters in accordance with AES17-1998 (section 4.2.1.1). It replaces the earlier Audio Precision S2-AES17LP filter.

The filter option consists of a dual-frequency pre-analyzer low-pass module, the SLPX; two additional analyzer option filters, FLP-B20K and FLP-B40K, and installation software and instructions.

An essential feature of the pre-analyzer low-pass module is its location previous to the Analog Analyzer, where it attenuates the out-of-band noise components before they can overload the circuitry and make distortion and other low-level measurements difficult. The additional option filters complete the job to satisfy the AES17 recommendation.

For specified performance, the S-AES17 option must be used with both the pre-analyzer low-pass module and the option filter FLP-B20K when set for 20 kHz, and with option filter FLP-B40K when set for 40 kHz.

S-AES17 Specifications	
Passband Response, 20 kHz setting:	± 0.10 dB, 10 Hz–20.0 kHz
Passband Response, 40 kHz setting:	± 0.10 dB, 10 Hz–40.0 kHz
Stopband Attenuation, 20 kHz setting:	≥ 60 dB, 24.0 kHz–200 kHz
Stopband Attenuation, 40 kHz setting:	≥ 60 dB, 48.0 kHz–200 kHz
Residual THD+N (1 kHz), 20 kHz AES mode: Residual THD+N (1 kHz), 40 kHz AES mode:	≤ (0.0003% [–110.5 dB] +1.0 μV) ≤ (0.0004% [–108 dB] +1.4 μV)

Appendix B

MATLAB Functions

MATLAB is a technical computing environment used in DSP design and analysis. MATLAB is a product of The Mathworks, Inc., who can be reached at 508-647-7000 or on the Web at http://www.mathworks.com.

The following MATLAB functions can be found in four MATLAB files (each with the filename extension *.m), which have been installed in the folder C:\Apwin\Matlab. The files are provided with APWIN version 2.11 and later versions, as an aid to our customers who also use MATLAB.

These functions are useful in creating and modifying downloadable filters and Audio Precision waveform files for use with APWIN and System Two Cascade.

Downloadable Filter Support

Two MATLAB functions supporting Audio Precision downloadable filters are provided. `ap_write_filter` generates filter files from within MATLAB, and `ap_read_filter` imports such files into MATLAB for further manipulation.

ap_write_filter

Syntax

```
size = ap_write_filter(filename,filter_type,info,sample_rate,sos)
size = ap_write_filter(filename,filter_type,info,sample_rate,b,a)
size = ap_write_filter(filename,filter_type,info,sample_rate,z,p,k)
size = ap_write_filter(filename,filter_type,info,sample_rate,a,b,c,d)
size = ap_write_filter(fid,filter_type,info,sample_rate,sos)
size = ap_write_filter(fid,filter_type,info,sample_rate,b,a)
size = ap_write_filter(fid,filter_type,info,sample_rate,z,p,k)
size = ap_write_filter(fid,filter_type,info,sample_rate,a,b,c,d)
```

Description

`ap_write_filter` generates a text file in the format recognized by the downloadable filters feature of APWIN version 2.11 and later. Filters in any of the standard MATLAB forms (second-order section, transfer function, zero-pole-gain, and state-space) can be written to the file, along with the sample rate and textual information.

The AP downloadable filter format consists of one or more sample rates, with accompanying filter coefficients in the form of second-order sections. `ap_write_filter` converts the filter into second-order section form before writing it to the file. This will require functions found in the MATLAB signal processing toolbox if the filter is not already in second-order section form.

`ap_write_filter` accepts both filenames and file handles. If a filename is supplied, any extension is replaced with .afl, .afh, or .afw, depending on the filter type. If the filename ends in '+', the file is appended to, or created if non-existent. `ap_write_filter` closes the file when the function exits. If a file handle is supplied, `ap_write_filter` appends to the file, leaving it open on exit.

The arguments `filter_type`, `info`, and `sample_rate` are required. `filter_type` is the string '`l`', '`h`', or '`w`', for low-pass, high-pass, and weighting filters respectively. `ap_write_filter` uses this information to determine the filename extension, and to restrict the number of second-order sections to the maximum allowed in the Audio Precision hardware for each filter type. `info` is a character string that can be viewed inside APWIN by clicking the **Filter Info** button after loading the filter. This provides filter information to the APWIN user. `sample_rate`, a scalar, is the sample rate in hertz for which the filter was designed. See **Multiple Sample Rates** for more information.

`ap_write_filter` uses the number of remaining arguments to determine whether the filter is supplied in second-order section form, in transfer function form, in zero-pole-gain form, or in state-space form. The transfer function form is not recommended for large filters because factorization into second-order sections may cause the poles and zeros to move slightly, deforming the filter response.

The total number of bytes written to the file is returned in `size`.

Multiple Sample Rates

A digital filter will have the intended frequency response only at the sample rate for which it was designed. Thus, a unique filter must be designed for each sample rate that will be encountered. If a filter file contains filters for multiple sample rates, APWIN chooses the filter coefficients corresponding to the sample rate closest to the current hardware sample rate.

An Audio Precision filter file can contain a filter defined at any number of sample rates. Either a file handle or the file append mode can be used to add filters for other sample rates to an existing file. `ap_write_filter` writes an **info** string for each sample rate; however, APWIN will only report the first of these strings.

Filter Restrictions

The number of filter orders is limited to 6 for low-pass filters, 4 for high-pass filters, and 8 for weighting filters. The sample rate must be between 6750 Hz and 262144 Hz. All zeros must be on or inside the unit circle, and all poles must be inside the unit circle. All coefficients must be in the range [-2,2].

Examples

1. Compute a Chebyshev high-pass filter with a corner frequency of 200 Hz and a passband ripple of 0.1 dB for three common sample rates.

```
for sr=[32000 44100 48000]
  [z,p,k]=cheby1(4,0.1,200/(sr/2),'high');
  ap_write_filter('c:\filters\cheby200+','h','200 Hz Cheby highpass',sr,z,p,k)
end
```

We use the zero-pole-gain form for accuracy, and the append mode to attach the three sample rates to the same file.

2. Compute a voice-band boosting filter at two sample rates. Use the file handle method to write to the output file.

```
ofp=fopen('c:\filters\voice.afw','wt');
fprintf(ofp,'# Voice band filter designed in MATLAB by yulewalk\n\n');
for sr=[48000 65536]
  [b,a]=yulewalk(8,[0 500 1000 4000 8000 sr/2]/(sr/2),[0.5 0.5 1 1 0.25 0.25]);
  ap_write_filter(ofp,'w','Voice band filter',sr,b,a)
end
fclose(ofp);
```

Here we have the flexibility to write comments to the output file (preceded by the '#' identifier) which will not appear inside APWIN. Note that it is the user's responsibility to create a file with the appropriate extension, and to properly close the file.

See also `ap_read_filter`.

ap_read_filter

Syntax

```
[filter_array,info] = ap_read_filter(filename)
[filter_array,info] = ap_read_filter(fid)
```

Description

`ap_read_filter` imports an Audio Precision downloadable filter text file into MATLAB. It parses the file and returns a structure array of filters, one structure for each sample rate contained in the file. It also returns the **info** comment strings. The file is closed on exit.

The AP downloadable filter format consists of one or more sample rates, with accompanying filter coefficients in the form of second-order sections. `ap_read_filter` creates one structure in the output array for each sample rate in the file. Each structure contains the sample rate and a matrix consisting of the filter in second-order section form. If other forms are required, such as state-space, the signal processing toolbox conversion functions should be used.

Comments in the file are discarded, but **info** strings are concatenated into the returned `info` variable. The **info** strings are used by APWIN to provide information to the user when the **Filter Info** button is clicked.

`ap_read_filter` attempts to deal with syntax errors in the file by issuing a warning of the line number of the error.

Example

To read the filter file generated by the first example in `ap_read_filter`:

```
[filters,info] = ap_read_filter('c:\filters\cheby200.afw');
```

This returns a 1×3 structure array. Typing `filters(1)` at the prompt returns

```
ans = sample_rate: 32000
              sos: [2x6 double]
```

The second-order sections can be examined by typing `filters(1).sos` at the prompt.

See also `ap_write_filter`.

AP Waveform File Support

Two MATLAB functions supporting Audio Precision formatted wave files are provided. `ap_write_wave` generates AP wave files from within MATLAB, and `ap_read_wave` imports such files into MATLAB for further manipulation.

ap_write_wave

Syntax

```
size = ap_write_wave(filename,sample_rate,normalize,data)
size = ap_write_wave(fid,sample_rate,normalize,data)
```

Description

`ap_write_wave` generates a binary waveform file that can be loaded into Audio Precision hardware. Files with extensions .agm (mono generator waveform) and .ags (stereo generator waveform) are supported.

The AP waveform file format consists of a 256-byte header containing sample rate and other information, and a 3*n*-byte payload, where *n* is the number of samples in the waveform. Each sample is quantized to 24 bits and is in the range $[-1, 1-2^{-23}]$.

`ap_write_wave` accepts both filenames and file handles. If a filename is supplied, any extension is replaced with .agm or .ags, depending on the size of the data matrix. `ap_write_wave` closes the file when the function exits. If a file handle is supplied, `ap_write_wave` appends to the file, leaving it open on exit.

`sample_rate`, a scalar, is the sample rate of the waveform in hertz.

If `normalize` is 1, the data is scaled so that the peak of the waveform reaches full scale. The data is also dithered before reducing the word size to 24 bits. This preserves low-level information and eliminates harmonic distortion due to quantization. This mode is recommended unless multiple waveforms are being generated whose amplitude relative to one another is important.

If `normalize` is 0, the data is written to the file unscaled. If any samples are outside the allowable range, they are clipped and a warning is issued.

`data` is a matrix containing the waveform. If the smaller dimension is 1, a mono waveform file is generated. If it is 2, a stereo waveform file is generated. Any other size will generate an error.

Example

At a sample rate of 40 kHz, create a sine wave of approximately 1 kHz in channel A and noise in channel B. The length is 2048 samples.

```
len=2048;
samp_num=0:len-1;
samp_rate=40000;
ch_A=sin(samp_num/len*2*pi*round(1000/samp_rate*len));
ch_B=randn(1,len);
ap_write_wave('c:\waveforms\sin_noise',40000,1,[ch_A; ch_B])
```

This generates a file sin_noise.ags. The actual frequency of the sine wave is 996.094 Hz. This is a frequency that is synchronous with the sample rate and the waveform length. That is, when the waveform is repeated, there are no discontinuities.

See also `ap_read_wave`.

ap_read_wave

Syntax

```
waves = ap_read_wave(filename)
waves = ap_read_wave(fid)
```

Description

`ap_read_wave` imports an Audio Precision waveform file into a MATLAB structure array. One structure is created for each waveform in the file. Files with extensions .aam (mono acquired waveform), .aas (stereo acquired waveform), .agm (mono generator waveform), and .ags (stereo generator waveform) are supported. The file is closed on exit.

Each structure consists of a sample rate, a trigger point, a time vector, and the waveform vector. The sample rate and trigger point are determined from the header information in the waveform file.

For acquired waveforms, the trigger point is the sample at which the trigger occurred during acquisition. For generated waveforms (such as arbitrary waveforms created by the Multitone Waveform utility in APWIN), the trigger point is usually zero. The time vector is constructed from the trigger point, the total number of points in the waveform, and the sample rate. The data vector consists of the file payload converted to `double`.

If `ap_read_wave` determines that the file contains data acquired from an analog source, it attempts to scale the data so that the value of a sample represents the analog voltage at that time. Because some gain constants are known only to the hardware on which the waveform was acquired, however, this will only be approximate. Data originally from a digital source is not scaled.

Example

Read the .ags file generated by the `ap_write_wave` example.

```
waves=ap_read_wave('c:\waveforms\sin_noise.ags');
```

This returns a 1×2 structure array. Typing `waves(1)` at the prompt returns

```
ans = sample_rate: 40000
    trigger_point: 0
             time: [2048x1 double]
             data: [2048x1 double]
```

The data can now be examined:

```
figure(1), plot(waves(1).time,waves(1).data)
figure(2), plot(waves(2).time,waves(2).data)
```

See also `ap_write_wave`.

Audio Precision
5750 Arctic Drive
Beaverton Oregon 97075
Tel: 503-627-0832 Fax: 503-641-8906
US Toll Free: 1-800-231-7350
email: info audioprecision.com
Web: audioprecision.com