

APWIN SIMPLIFIED



TUTORIAL, MULTITONE & PROCEDURES
FOR SYSTEM TWO

APWIN Simplified System Two



Tutorial

Getting Started with Multitone Testing

Getting Started with Procedures

Version 1.5
December, 1997

Copyright © 1997 Audio Precision, Inc.

All rights reserved

Version 1.5 December, 1997

APWIN Basic and APWIN Basic Editor
Copyright 1995-1997 Audio Precision Incorporated
Copyright 1993-1995 Polar Engineering and Consulting
All rights reserved.

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

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

Published by:



Audio Precision, Inc.
PO Box 2209
Beaverton, Oregon 97075-3070
U.S. Toll Free: 1-800-231-7350
Tel: (503) 627-0832 Fax: (503) 641-8906
email: techsupport@audioprecision.com
Web: www.audioprecision.com

Printed in the United States of America
Audio Precision Part # 8211-0014

Contents

Book One

APWIN Tutorial: System Two	1
Introduction	1-1
Overview	1-1
Conventions	1-2
System Two's Toolbar	1-3
Levels of Capability	1-4
Major Options and Accessories	1-7
Conceptual Block Diagram	1-8
Generator Architecture	1-8
Analyzer Architecture	1-9
System Two	1-12
Signal Connectors	1-12
APWIN Panels	1-14
Settings and Readings	1-17
Setup "from Scratch" vs Opening Previously-Saved Tests	1-18
Loudspeaker-Headphone Operation	1-20

Modes of Operation	2
Impromptu Real-Time Measurements	2-1
Sweeps Providing Sets of Real-Time Measurements	2-4
DSP Batch-Mode Operation with Time or Frequency Domain Measurements	2-19
Multitone Testing	2-23
Procedures (“Macros” or “Scripts”) Comprised of Sequences of Operations	2-24
Equalization Functions	2-25
Regulation Function	2-27
Displaying Sweep Results	2-29
Zooming	2-31
Optimizing	2-34
Cursors	2-35
Graph Title and Labels	2-37
Printing Graphs	2-39

Book Two

Introduction To Multitone Testing	1
Concepts	2
How Multitone Testing Works	2-1
Response Measurements	2-3
Crosstalk Measurements	2-5
Phase Measurements	2-7
Distortion Measurements	2-8
Noise Measurements	2-10
Hum Measurements	2-12
Sequence of Events	2-13
Introduction to Setup Chapters	3

Analog Multitone Testing with System Two	4
Flow Diagram	4-1
Setup Overview	4-4
Tutorial - Generating a Multitone	4-13
Tutorial - Generator Settings	4-14
Tutorial - Analog Analyzer Settings	4-16
Tutorial - Digital Analyzer Settings	4-17
Tutorial - Graph Setup	4-20
Digital Multitone Testing with System Two	5
Flow Diagram	5-1
Setup Overview	5-4
Tutorial - Generating a Multitone	5-11
Signal Routing	5-12
Tutorial - Digital Analyzer Setup	5-14
Tutorial - Graph Setup	5-15
Advanced Concepts	6
FFT Length	6-1
Frequency Resolution	6-3
Triggering	6-4
Trigger Delay	6-7
FFT's: The Looping Problem	6-8
Windows	6-10
Synchronous Processing	6-12
Multiple Measurements on the Same Data	7
Multiple Measurement Concepts	7-1
Automating Multiple Measurements	7-3
Multiple Measurement Tutorial	7-4
Generating Custom Waveform Files	8
Sample Rate	8-1
Waveform Length	8-2

Frequency Considerations 8-3

Amplitude Considerations 8-5

Phase Considerations 8-7

Signal Triggering Considerations 8-9

Creating the ADX file 8-11

Running MAKEWAV2 8-12

MAKEWAV2 Command Line Options 8-14

APPENDIX A - Multitone Program Features 9

Book Three

Introduction i

Getting Started 1

 What Is a Program? 1-1

 Loading the Samples 1-2

 Using Learn Mode to Automate Tests 1-3

 What If I Make a Mistake? 1-8

 Adding to A Program 1-8

Using the Editor 2

 If You are New to Text Editing 2-1

 The Numbers on the side of the Editor Window 2-3

 Play, Pause, and Stop Buttons 2-3

 Reading Programs 2-4

 When is a program just like another? 2-7

 Aligning Your Program 2-9

 Making A Program Portable 2-10

 Creating Prompts 2-11

Log File Options	3
What is a Property?	3-1
Log File Properties	3-2
Example of Log Options	3-4
Re-usable Program Pieces (Procedures)	4
A Simple Procedure	4-3
Building a Library of Procedures	4-5
Other Hints	5
Comments	5-1
Don't be afraid to experiment!	5-3
Creating Reports	6
Including an Audio Precision Graph in your Reports	6-2
Changing the Appearance of your Graph	6-8
Going Further - More Graphs in a Report	6-9
Including Numerical Data in your Reports	6-10
Menus	7
The User Dialog Editor	7-2
A Simple Menu	7-4
Going Further - Restarting the Menu each time	7-8
Multiple Menus in One Program	7-14
Using Check Boxes in your Dialog Boxes	7-16
Changing your Check Box's Default	7-18
Expanding the Select...Case Idea	7-20
More on the If...Then	7-21

APWIN Simplified for System Two

Book One



Tutorial

Getting Started with Multitone Testing

Getting Started with Procedures

APWIN Tutorial: System Two

Introduction

This booklet is designed to lead you through your first operating session with APWIN and System Two. It assumes that you have installed APWIN software and the interface card in your computer and connected the interface card to System Two as described in the “Getting Started” booklet.

It is intended that you go through this tutorial process in the sequence in which it is presented. Most sections build on the skills learned in earlier sections. The booklet contains many exercises to give practical illustration of the concepts. Most of these exercises use test setups stored as disk files. *No external equipment or cables are needed for the exercises **and** no cables should be connected to System Two’s analog outputs while the Tutorial is in use.*


For more detailed information on operating APWIN and System Two, refer to the User’s Manual.


Overview

System Two is an audio test set controlled by a personal computer. It has broad, high-performance capabilities for analog, digital, and mixed-signal devices. System Two includes both signal generation and analysis capability for audio stimulus-response testing. Virtually all common and many specialized tests are performed on analog domain and digital domain signals and on the digital interface signal (pulse train) itself. APWIN’s primary orientation is toward sweep tests which provide sets of data with graphic representation of the test results. It can also provide tabular display instead of or in addition to graphs. System Two and APWIN are easily used in an impromptu fashion for single “spot” measurements with alphanumeric or simulated analog meter display.

Conventions

The following graphical and typographical conventions are used in this Tutorial:

 This graphic indicates a procedure for operating APWIN under Windows 3.11 which differs from that of Windows 95 APWIN operation.

 This graphic indicates a procedure for operating APWIN under Windows 95 which differs from that of Windows 3.11 APWIN operation.



Note:

All note paragraphs begin with the Pencil icon and contain important information for using this tutorial.



Exercise:

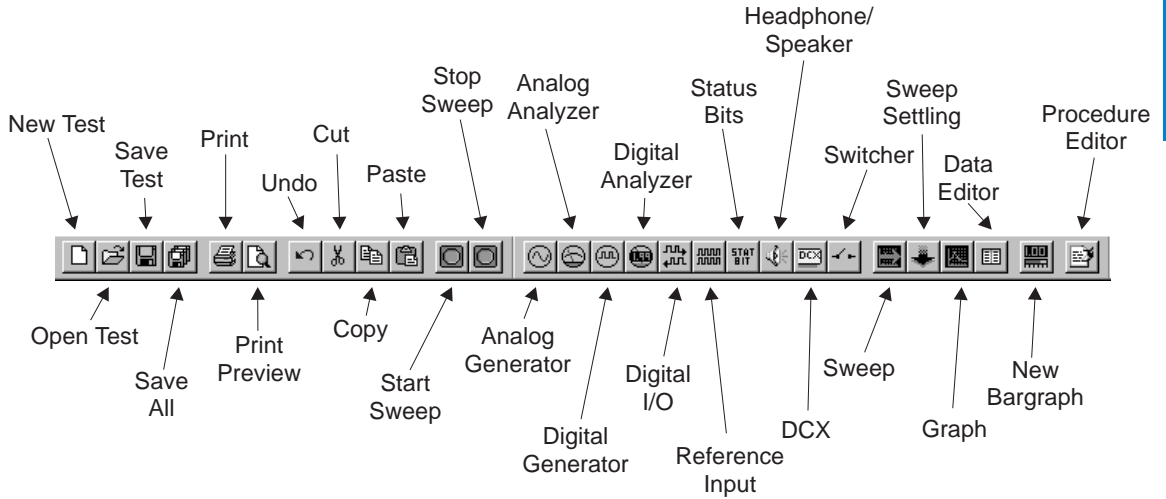
All exercises begin with the Pencil and Paper icon and provide you with the opportunity to “see in practice” the preceding topic.

DOS paths and filenames use the Courier typestyle as follows:

C:\APWIN\S2\TUTORIAL\APWIN\TIME_SWP.AT2

Each major section of the tutorial is indicated by the “Thumb Tabs” on the outer edges of each page and at the top inside of each page. The subsections are shown at the top outside of each page.

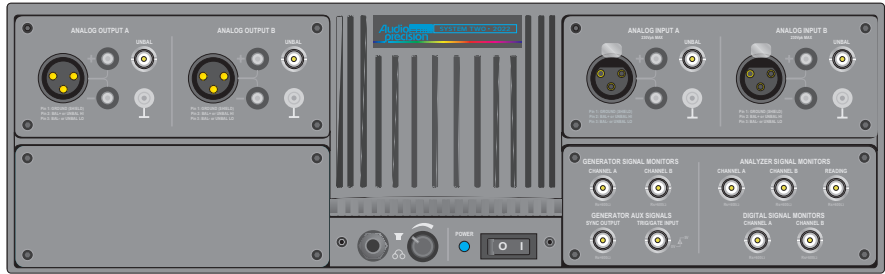
System Two's Toolbar



Levels of Capability

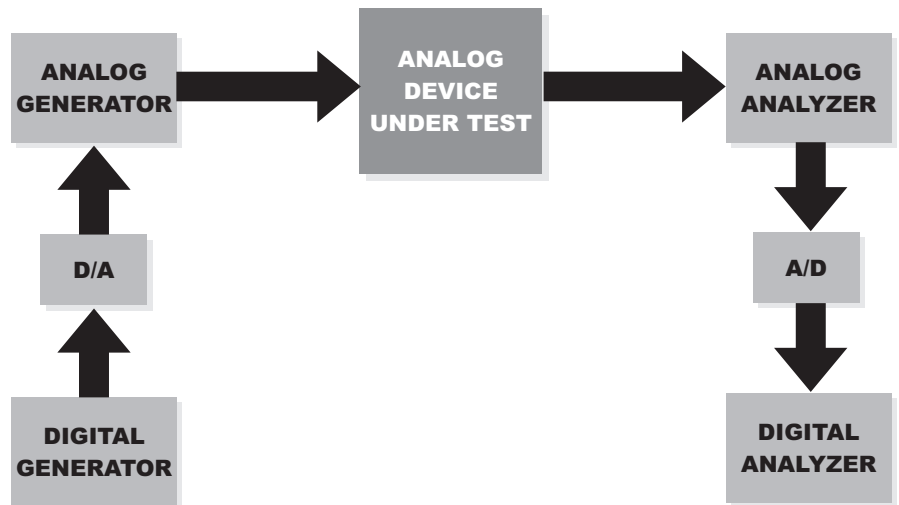
System Two is available in three progressively-increasing levels of capability:

SYS-2022 Uses analog hardware to test analog domain devices



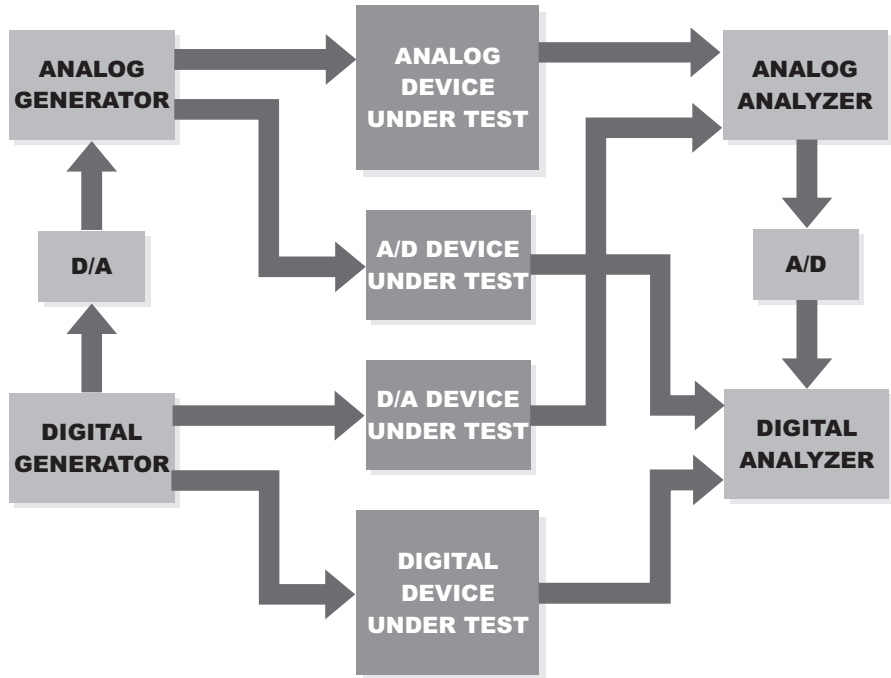
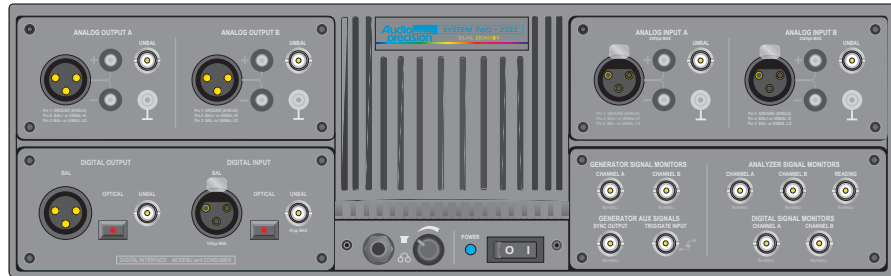
SYS-2222

Adds DSP-based module for advanced tests of analog domain devices (FFT spectrum analysis, digital storage oscilloscope operation, multitone testing)



SYS-2322

Adds digital input/output module for tests of digital domain devices and the serial digital interface signal itself



Major Options and Accessories

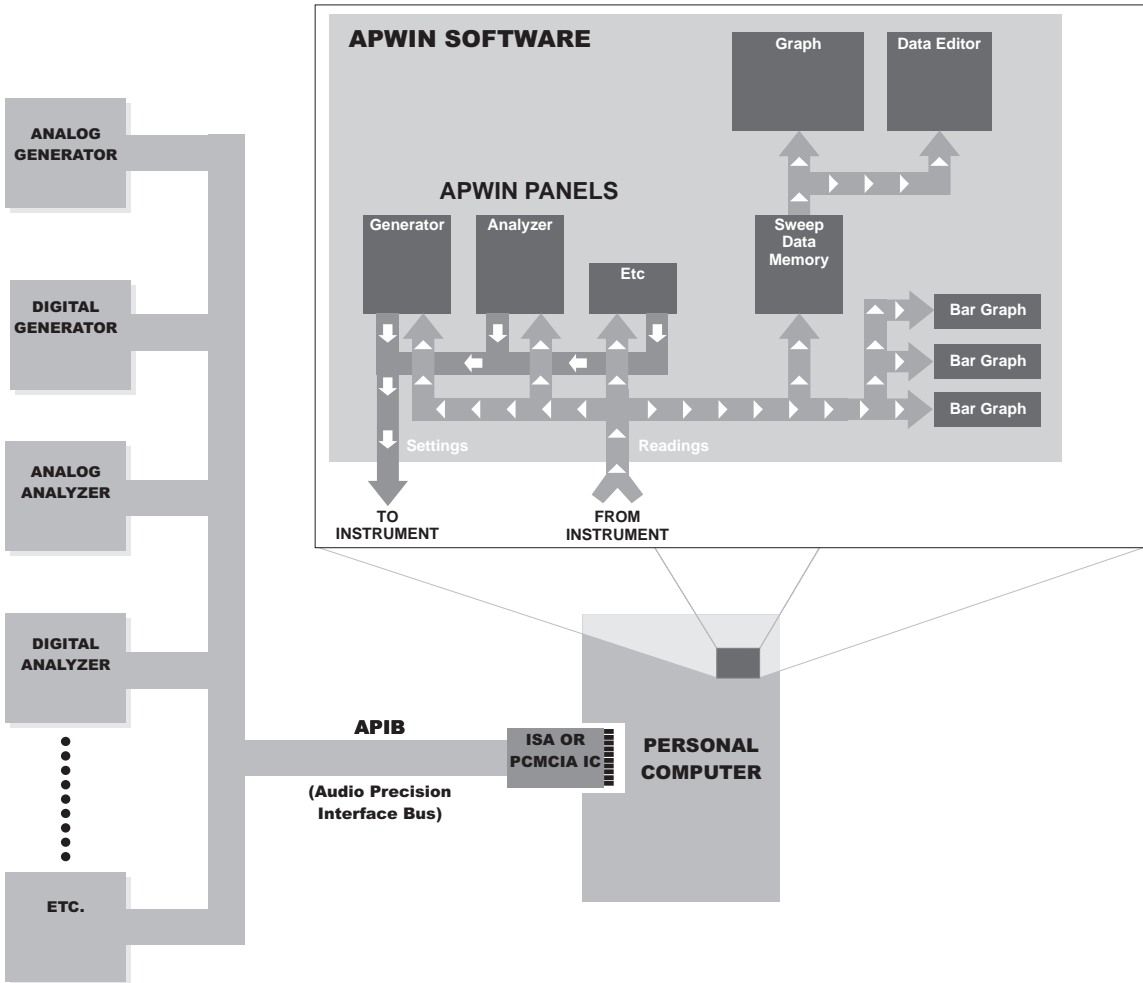
Several major options and accessories may be added to any of the three levels of System Two to permit specialized measurements on specific types of devices. The **three major options mount internally** in System Two and are designed for use only with analog devices under test:

- | | |
|-------------------|---|
| S2-IMD | Provides stimulus-response intermodulation distortion testing according to three standards (SMPTE/DIN, CCIF/DFD, DIM/TIM). |
| S2-W&F | Measures wow and flutter |
| S2-BUR | Generates special signals including squarewave, sinewave bursts, and random or pseudorandom noise of white, pink, or narrowband characteristics |

The **three major accessories are separate units** in single rack unit cabinets:

- | | |
|------------------|--|
| Switchers | 2 x 12 channel relay switching matrix for analog signals, available in input, output, and insertion (patch point) versions. Controlled from APWIN software by “daisy chain” connection to Audio Precision Interface Bus (APIB). |
| DCX-127 | Multifunction interface unit which measures DC voltage and resistance, generates two variable high-precision DC voltages, measures and generates 21-bit low-speed digital words, and provides a number of digital output lines. Controlled from APWIN software by “daisy chain” connection to APIB. |
| SIA-2322 | Serial interface adapter to convert System Two’s 24-bit parallel input and output to a wide variety of serial digital interface formats typical of chip level and module level interconnects. The SIA-2322 is not required for AES/EBU or SPDIF/EIAJ operation, which is standard in the SYS-2322. Format selection is controlled from front panel switches. |

Conceptual Block Diagram



Generator Architecture

The primary **System Two** series (SYS-2022, SYS-2222, and SYS-2322) all include the basic analog generator. The analog generator consists of an ultra-low distortion sinewave oscillator which drives two identical power amplifiers, fine amplitude control circuits, output transformers, and switched attenuators. Amplitude is thus individually controllable on each channel, but the same frequency

signal is present at both. The IMD (intermodulation distortion) option, if present, adds the ability to generate either a low-frequency sinewave with or without a balanced modulator circuit or a squarewave of approximately 3 kHz. These signals, combined with the sinewave from the low distortion oscillator, create the SMPTE/DIN, CCIF/DFD, and DIM/TIM test signals. The BUR option, if present, adds the ability to generate a variable-frequency squarewave, bursts of sinewaves from the main oscillator, and random or pseudo-random noise of white, pink, or narrowband pink spectral distribution. All of these signal generating options drive both output channels with identical waveforms.

On **System Two + DSP** models (SYS-2222 series), in addition to the analog signal generation capability, a DSP-implemented digital generator and stereo D/A converters permit digitally-generated waveforms instead of the analog signal sources described above to be fed to the analog power amplifiers and output channels. Most digitally-generated signals drive both analog output channels simultaneously with the same waveform. Two of the digitally-generated waveforms, Stereo Sine and Arbitrary Waveform, can feed independent signals to the analog generator channels A and B output circuits.

With **System Two Dual Domain** (SYS-2322 series), in addition to all the capability described above, a DSP implemented digital generator can drive direct digital outputs in any of several formats.

Analyzer Architecture

The primary **System Two** series (SYS-2022, SYS-2222, and SYS-2322) all include the basic analog analyzer. It is capable of measuring level and amplitude (wideband, band-limited, selective, or through one of a wide variety of available plug-in optional weighting filters), frequency, phase, and THD+N (total harmonic distortion plus noise). Two identical input channels include balanced and unbalanced inputs, amplifiers and switchable attenuators, and a Level meter (RMS detector) and Frequency counter on each channel. A Phase meter measures phase difference between the two input channels. Either input channel may also be connected to a more flexible meter often referred to as the Reading meter or main meter. This Reading meter

offers a selection of five types of detector response—RMS, average, peak, quasi-peak, and sine-scaled peak. Three high-pass filter selections are available (22 Hz, 100 Hz, 400 Hz) to limit the low-frequency bandwidth which otherwise extends below 10 Hz. Three low-pass filter selections are available (22 kHz, 30 kHz, 80 kHz) to limit the high-frequency bandwidth which otherwise extends above 500 kHz. Any of seven internal filter sockets may be selected, which may have optional weighting filters, bandpass filters, or high-pass or low-pass filters plugged into them. A tunable bandpass-bandreject filter is also selected in several of the functions of the Reading meter. In Bandpass and Crosstalk functions, the filter is a 1/3 octave bandpass filter which rejects wide-band noise and signals away from the center frequency. In Bandreject, THD+N Relative, and THD+N Absolute functions, the filter is a sharp bandreject configuration which rejects a narrow band of frequencies while passing the remainder of the audio spectrum. If the IMD (intermodulation distortion) option is present, its three different IMD measurement standards are selectable as functions of the Reading meter. If the W&F (Wow and Flutter) option is present, it is also selectable as a function of the Reading meter. All six measurement meters (Level A, Level B, Frequency A, Frequency B, Phase, and the Reading meter in whichever function is selected) function simultaneously and independently.

The **System Two + DSP** series (SYS-2222) adds DSP-based analysis and generation of signals to the capabilities described above. The DSP analyzer is a two-channel design. At the user's choice, high-resolution (20-bit) A/D converters with 24 kHz bandwidth or wide-bandwidth (80 kHz) A/D converters with 16-bit resolution may be selected. The A/D converter inputs may take signal from either the input channels A and B, following input ranging and balanced-to-unbalanced conversion, or from the Reading meter following all filtering. Waveform display and spectrum analysis of the selected signals is performed by the FFT Spectrum Analysis DSP program (FFT.AZ2). Synchronous FFT analysis and post-processing for very rapid audio device testing and codec testing via multitone technology are provided by the Multitone Audio Analyzer DSP program (FASTTEST.AZ2). Very fast response sweeps can be made with the DSP Audio Analyzer program (ANALYZER.AZ2). The Quasi-Anechoic Acoustical Tester (MLS.AZ2) program performs

specialized acoustical measurements. The waveforms selectable on the Analog Generator panel as “(D/A)” are created in a DSP-implemented generator, converted to analog signals by 18-bit stereo D/A converters, and fed to the twin power amplifiers and output channels of System Two’s analog generator. This DSP-implemented generator is a two-channel design that can generate a wide variety of waveforms in real time or can generate arbitrary waveforms from a down-loaded waveform file.

The **System Two Dual Domain** (SYS-2322) series adds digital domain input and output capability in AES/EBU and SPDIF (Consumer) formats at XLR, BNC, and optical input and output connectors plus general-purpose serial and parallel inputs and outputs. The parallel inputs and outputs may further be connected to the SIA-2322 Serial Interface Adapter which is user-configurable via front-panel switches to a wide variety of proprietary serial formats including the Philips I²S. A reference input/output panel on the rear of the instrument is also present in Dual Domain instruments. This reference capability permits synchronization of System Two’s digital generator output to AES/EBU, squarewave, or video inputs and allows measurement of the timing of front-panel digital inputs relative to the reference input. The reference output signal may be used as “house sync” to other digital devices, and the front-panel digital generator output timing may be varied with respect to the reference output. Analysis capabilities of the Dual Domain unit for digital input signals include all the waveform display, spectrum analysis, and synchronous multitone analysis described above. In addition, Dual Domain units can make real-time measurements of wideband, selective, and weighted level, frequency, THD+N, level ratio, and crosstalk on digital input signals via the DSP Audio Analyzer (**ANALYZER . AZ2**) program. Furthermore, Dual Domain units measure all the major characteristics of the AES/EBU or SPDIF digital input serial pulse train (physical interface signal) itself. These include pulse amplitude, sample rate, rise and fall times, jitter, normal mode noise, common mode noise, and delay from reference. Some of these measurements are available at all times and others, implemented by a built-in 67 MHz sampling oscilloscope, are available through the Digital Interface Analyzer DSP program (**INTERVU . AZ2**). The AES/EBU or SPDIF digital output may have a wide variety of deliberate, calibrated

impairments added to it in order to test the tolerance of device digital inputs. These impairments include variable sample rate, variable pulse amplitude, variable rise and fall times, addition of normal mode noise and common mode sinewave interference, simulation of a long cable, and addition of variable amounts of jitter of several waveforms.

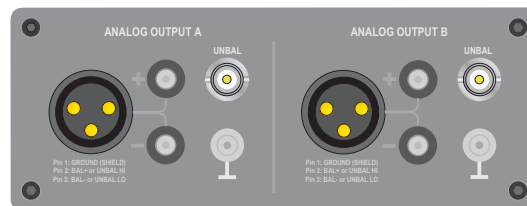
System Two



Each connector panel section is explained below.

Signal Connectors

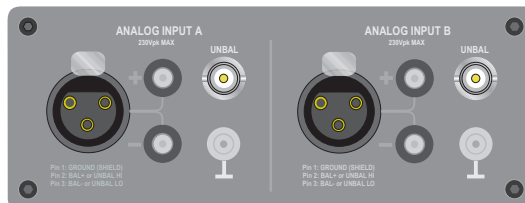
Upper-Left Connector Panel



The **upper-left** connector panel contains all of System Two's analog generator output connectors, with Channel A at the left and Channel B at the right. APWIN software panels, described below, permit driving A only, B only, or both simultaneously with both in phase or one inverted with respect to the other. The XLR and double-banana connectors are hard-wired in parallel and thus both are

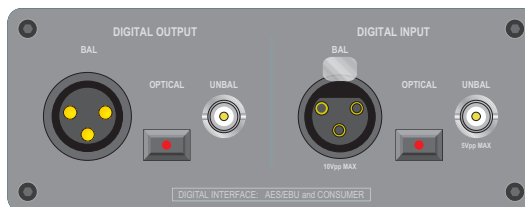
driven in balanced mode. In unbalanced mode, the BNC connector is also driven.

Upper-Right Connector Panel



The **upper-right** connector panel contains all of System Two's analog analyzer input connectors, with Channel A at the left and Channel B at the right. APWIN software panels permit selection of either the BNC or the parallel-wired XLR and banana jacks independently on each channel. The selected Channel A connector always drives the Level A and Frequency A meters and one input of the Phase meter. The selected Channel B connector always drives the Level B and Frequency B meters and the other input of the Phase meter. APWIN software panels permit selection of whether the Channel A or Channel B input connector drives the Reading meter.

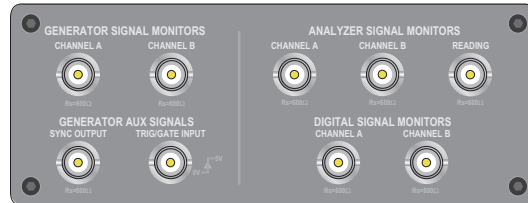
Lower-Left Connector Panel



A **lower-left** connector panel is provided with Dual Domain (SYS-2322) units. Three output connectors are grouped at the left and three input connectors at the right. XLR balanced, BNC unbalanced, and optical (Toslink) connectors make up each group. All three digital output connectors are driven if any one of them is selected in the Output section of the DIO (Digital Input Output) software panel. Only one of the digital input connectors may be used at any one time, as

selected in the Input section of the DIO panel. General-purpose serial and parallel input and output connectors are located on the rear panel of digital i/o versions of System Two.

Lower Right Connector Panel



The **lower-right** connector panel provides a number of BNC connectors which permit monitoring of key generator and analyzer signal points on an external oscilloscope or spectrum analyzer, plus one BNC input for control of the sine burst option.

APWIN Panels


Panels are *individual* windows within APWIN which provide control and display of a group of related features of System Two. For example, the Analog Generator panel includes all the settings which control the analog signal output functions of System Two. The Analog Analyzer panel includes all settings and readings of the analog measurement functions. Readings are displayed on panels as green numerals against a black background. They update continually as new measurements are made. Settings may be numeric entry fields, drop lists with multiple-choice selections, checkboxes, OFF/ON (toggle) buttons, “radio buttons” to select only one from several choices, and other typical Windows control techniques.



Note:

No cables should be connected to System Two’s generator outputs during any of the Exercises described below.


 **Exercise:**


Click the **new test icon** . At the Analyzer Channel A Source field, select GenMon (generator monitor); you should hear a relay click in the Analyzer. This connects via internal cable the generator Channel A output to the Channel A analyzer input. Turn the Analog Generator on by clicking the large OUTPUTS button in the lower center of the generator panel. The button label will change to ON and a green indicator will light, and you should hear several relays click as the Analyzer autoranges to the correct level. The Analyzer Level meter should indicate almost exactly 1.000 Volt and the Analyzer Frequency counter should indicate almost exactly 1.000 kHz. If you make changes to the Analog Generator Frequency or Channel A Amplitude fields, the new values (within a very small tolerance) should be displayed by the Analyzer Channel A meters.

Panels continue to control their portion of the instrument whether or not they are displayed.

 **Exercise:**

Kill the Analog Generator panel by:

311 Win 3.1: double-click the window's control box  in the upper left corner of the Analog Generator panel, or single-click the control box and select Close from the list of commands which appears

95 Win95: click the "X" icon  in the upper right corner of the Analog Generator panel.

Note that the Analyzer readings don't change since the Analog Generator is still operating under the same conditions even though the panel is no longer visible.


Any panel may be brought to the screen by either of two techniques. The Panels command of the menu displays a list of all available panels. Selecting one of them displays that specific panel.

Alternately, the Panels Toolbar 

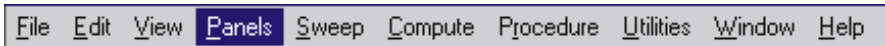
includes an icon for each available panel so a panel may be displayed by clicking on the icon.



Exercise:

Click the new test icon .


Kill both the Analog Generator and Analyzer panels. Display the **Sweep** panel by clicking the Panels command in the Menu bar




then click the Sweep selection from the list of panels which will be displayed. Display a Bargraph by clicking the **New**

Bargraph icon .

Most panels may be displayed in both small and large versions. The small version shows only the most commonly-used settings and readings fields, while the large version shows all fields. Switching between large and small is accomplished by:


311 Win 3.1: double-clicking anywhere in the Title Bar of the panel, or single-clicking the icon  in the upper right corner of each panel

95 Win 95: double-clicking anywhere in the Title Bar of the panel, or single-clicking the center icon  of the three icons in the upper right of each panel.

A panel may be initially displayed in its large size by holding down the Shift key while clicking the toolbar icon.




Exercise:

Kill all panels presently on screen. Hold down the Shift key and click the **Sweep Panel** icon  to immediately display the large version of this panel.

When fields on a large panel are set and the panel is then reduced to the small size, those now-invisible fields still continue to control the instrument. In order to allow more space on the screen, it may be

desirable not to display a panel or display it in small size, increasing to large size only when necessary to alter rarely-changed settings.

Panels may be displayed on any of the *five* pages of APWIN workspace. Pages are selected *by clicking on the page tabs* near the bottom of the screen , or using the Ctrl-PgUp and Ctrl-PgDn keys. When APWIN software is first started, the Analog Generator and Analog Analyzer panels are displayed on page 1, the **Sweep Panel** and Graph on page 2, and the Digital Input/Output panel on page 3. The page number on the page tab is in bold text if any panels are displayed on that page. A panel may be displayed simultaneously on more than one page if desired. When multiple versions of the same panel exist they will all have the same effect.



Exercise:

Click on the New Test icon for a fresh start. Go to Page 4 and display several panels by clicking on their icons on the Toolbar. Go to page 5 and display the Analog Generator panel. Change the Amplitude and Frequency settings of the Analog Generator and turn its output ON. Go to page 1 and note that the Analog Generator settings there have also changed since all versions of the same panel are functionally connected in parallel.

Settings and Readings

Settings are the way that APWIN panels control the test instrument conditions. They function similarly to the knobs, buttons, and switches on conventional instruments. Settings may be:

1. a numerical value entry from the keyboard such as generator frequency or amplitude or analyzer bandpass filter frequency
2. a selection from a list of choices such as generator waveform (sinewave, squarewave, etc.) or analyzer high-pass or low-pass filter frequency
3. a binary (two value) selection such as generator output (on/off) or analyzer reading meter channel selection (A/B)



Exercise:

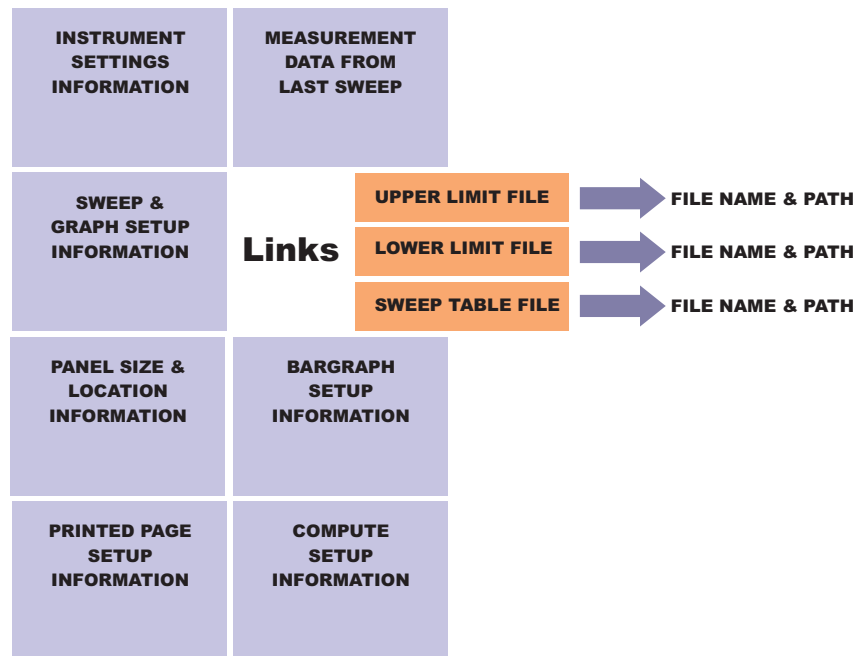
Click on the New Test icon. Set the Analog Generator Channel A Amplitude to 10 Volts. Set the generator frequency to 50 kHz. Turn the generator ON. Select GenMon as the Analyzer Channel A Source. Note that the Channel A Level, Channel A Frequency, and Amplitude meters should now correspond to the generator settings within a small tolerance. Click the down arrow at the right of the Channel A Level display field and select dBu units. The value displayed should be very close to +22.2 dBu. Click the down arrow at the right of the Amplitude meter display field and change the units to dBV. The value displayed should be very close to +20 dBV. Click the down arrow at the right of the Analog Generator Channel A Amplitude field and change to Vpp (Volts peak-to-peak) units. The Generator Amplitude display will become 28.28 Vpp. Ten Volts rms, +22.2 dBu, +20 dBV, and 28.28 Vpp (for a sinewave) are all the same value, simply stated in different units.

Readings are the display of measurement results. APWIN displays real-time numeric readings on panels or bar graphs which simulate analog meters. A series of two or more readings may also be displayed as a line on an X-Y graph. Readings function similarly to the analog meters, digital numeric displays, and CRT or LCD displays of conventional instruments. Examples of readings include level, distortion, noise, phase, frequency, sample rate, and jitter.

Setup “from Scratch” vs Opening Previously-Saved Tests

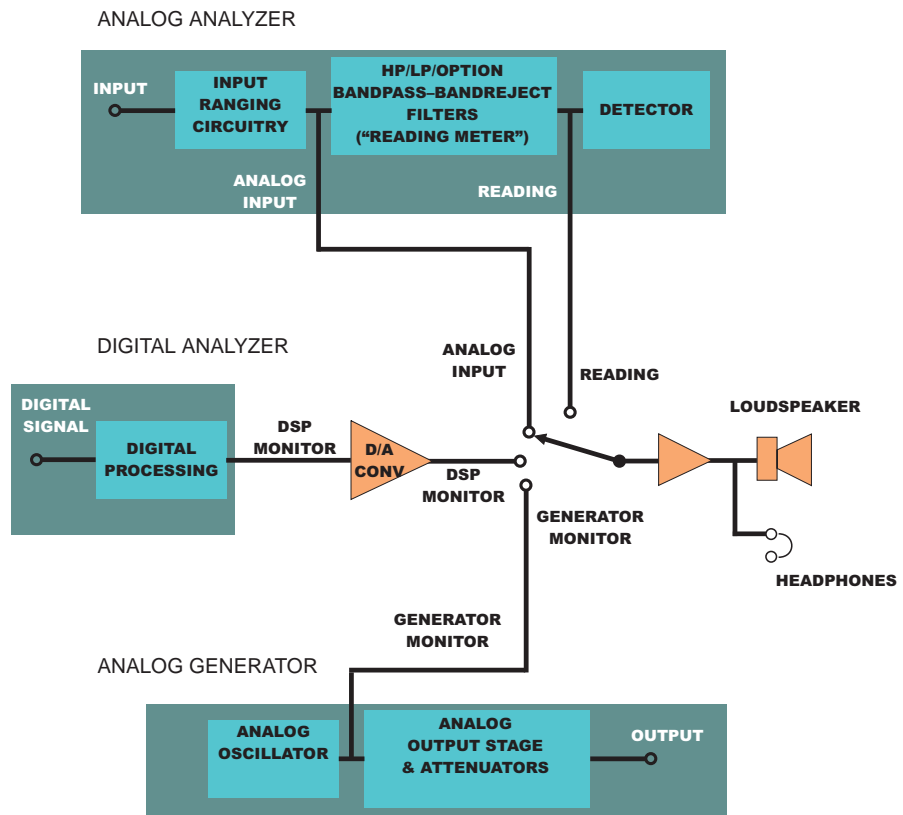
Obtaining a useful, reliable measurement of an audio device requires that many settings be properly made. Still more correct settings are required if a graph of a set of measurements (sweep) is desired. It is possible to create each setup by starting from the default start-up conditions of the software or from whatever present settings may exist. However, it is much faster, more reliable, and less error-prone to instead open a previously-saved test which has all the proper conditions already stored in it. Opening previously-saved tests assures standardization of testing methods from day to day and user to user.

Audio Precision furnishes a large number of typical, standard test files whose description fills the last chapter of the User’s Manual. These test files may be used (with modifications as necessary) to perform many standard audio tests. **If a furnished test is modified, it should be saved under a new name. It is strongly recommended that when any test setup has been made or modified which may ever be desired again, the Save As Test command be used and a name supplied in order to save this entire setup to disk for possible later use.**



Test file Architecture

Loudspeaker-Headphone Operation




Simplified diagram shows only one connection per functional block. In fact, all monitoring points are stereo except reading meter. Headphone connector is stereo. Both stereo channels are summed into a single loudspeaker.

The built-in loudspeaker, stereo headphone connector, and volume control permit audible monitoring of a number of key analog and digital signals in the instrument. The points which may be monitored include both analog analyzer input channels (Analog Input A, Analog Input B), the analog analyzer signal following all filtering (Reading), the analog generator output signal (Generator Monitor A, Generator Monitor B), and DSP module signal points (on DSP-equipped instruments) which depend upon the digital analyzer mode (DSP Monitor A, DSP Monitor B). Digital signals are converted to the analog

domain via stereo D/A converters dedicated to the audible monitoring function. The audible level will remain approximately constant over a wide range of signal levels due to the autoranging action of the analyzer. For more information, see the “Speaker/Headphone Panel” section of the “APWIN Menus” chapter of the User’s Manual.

**Exercise:**

Click the New Test icon. Display the Headphone/Speaker panel by clicking its Toolbar icon . Select Reading instead of Off for the signal source; Reading is the signal following all filters in the analog Analyzer. Turn the Analog Generator output ON and select the GenMon Source of the Analyzer. Adjust the volume control for a comfortable level. Try entering different values in the Analog Generator Channel A Amplitude field and note that there is little difference in the loudspeaker loudness at Generator levels from 25 Volts down to 5 microvolts due to the wide-range autoranging of the Analyzer.

1 Introduction

Modes of Operation

System Two and APWIN software may be operated in four general styles, depending upon the measurement needs and the degree of structure which is appropriate. These styles are:

1. Impromptu Real-Time Measurements
2. Sweeps Providing Sets of Real-Time Measurements
3. DSP Batch-Mode Operation for Spectrum Analysis or Waveform Display
4. Procedures (“Macros” or “Scripts”) Comprised of Sequences of Operations

Each of these modes of operation is discussed below.

Impromptu Real-Time Measurements

Impromptu testing is typical of activities such as trouble-shooting, alignment, general investigations of the operation of a device, and other similar unstructured or semi-structured testing. The instrument user may not initially have a clear goal or end result in mind, and the results of one measurement often lead to the selection of the next measurement to be made. The user may decide to change stimulus waveforms, frequencies, or amplitudes, to inject signal at different points in the device, or to measure results at different points in the device. If the values of results are important, they are either simply remembered or may be written down on paper by the user. This style of operation is the only style available on manual, non-computer-controlled instruments. System Two, even though computer-controlled, may also be operated in this fashion.


Most readings in System Two are a continuing stream of real-time measurements of analog or digital domain parameters such as level, distortion, noise, jitter, etc. These readings represent the value of the parameter at the moment of measurement. Readings are typically updated at rates ranging from about once per second to as often as 128 times per second, depending upon the particular measuring device and its settings. During unstructured, impromptu operation of System Two and APWIN, this continuing flow of real-time measurements may be observed in two fashions—on the numeric

display fields (green digits on a black background) of the instrument panel, or on a simulated analog meter display called a Bar Graph.



Exercise:

311

Use the File Set Working Directory command and select the directory **C:\APWIN\S2\TUTORIAL\APWIN** as the current working directory. Click on the OPEN TEST icon or use the File Open Test command; the contents of the TUTORIAL directory should be displayed. Open Test **BARGRAPH.AT2**. This bar graph displays the reading from the Analog Analyzer Reading meter, which is set to Amplitude mode. Note that the bar graph display units are dBV (decibels relative to 1.000 Volt) while the numeric display on the Analog Analyzer panel is presently in Volts units. The panel meter units and the bar graph units are independent and either may be chosen from a wide selection. Click on the Setup button  on the bar graph panel to display the Bar Graph Setup dialog. Click on the down arrow at the end of the “Left” field in the Axis area at the bottom of the dialog and select the Volts unit (which may be nV, μ V, mV, or V depending upon the numeric value). Click on the Lin button, and click OK. The bar graph should now be calibrated in Volts, linearly from 0 to 10 Volts. The large numeric readout should exactly agree with the Analog Analyzer panel numeric readout. Click on the Setup button again, change from Lin to Log, and click OK. The bar graph should now have a logarithmic scale from 100 nV at the left to 10 V at the right.

Settings may affect the value of readings. For example, measured noise level normally depends upon the specific filters selected in the analyzer. During stimulus-response measurements, where a generator provides signal to the input of the device while an analyzer measures the resulting output, the generator settings will also affect the readings.



Exercise:

Click on the large OUTPUT button in the lower center of the Analog Generator panel to turn the generator output off. The panel and bar graph should now display the residual noise level of the analyzer of less than one microvolt.

When making impromptu measurements, it is often necessary to adjust a stimulus parameter such as generator frequency or amplitude in order to achieve a certain target measurement value. Stimulus parameter settings may be adjusted by typing a new number into the generator control field. Settings may also be adjusted by a “Settings Bar Graph”, where the bar graph becomes an analog-like “fader” control connected to a numeric setting field, rather than a display from a reading field. For more information on Bar Graphs, see the “Bar Graphs” section of the “Graphs and Printing” chapter of the User’s Manual.

**Exercise:**

Open Test **BAR_TWO.AT2**. Note that this test includes a second bar graph below the measurement bar graph. The lower bar graph is a Settings bar graph connected to the Analog Generator Channel A amplitude field. Settings bar graphs are distinguished by the “fader” control which can be moved in a horizontal “track” instead of the magenta-colored analog indicator of the Readings bar graph. Drag the fader control by holding down the left mouse button, or click on the fader and then use the horizontal arrow keys to vary the Analog Generator amplitude. As you move the fader, note that the Analog Generator Channel A Amplitude field numeric value changes exactly with fader movement. Note the corresponding changes in measured values from the analyzer, displayed both on the Readings bar graph and on the Analog Analyzer panel.

Sweeps Providing Sets of Real-Time Measurements

Sets of measurements, usually graphed, are typically required by many more-structured applications which are part of engineering development, manufacturing test, and regular preventive maintenance programs. A series of real-time measurements (readings) are normally graphed in sequence as vertical (Y) axis variations, as a function of some independent variable plotted against the horizontal (X) axis. Typical independent variables are generator settings such as frequency or amplitude, or time (for chart recorder-style graphs). This fashion of operation is called a sweep.

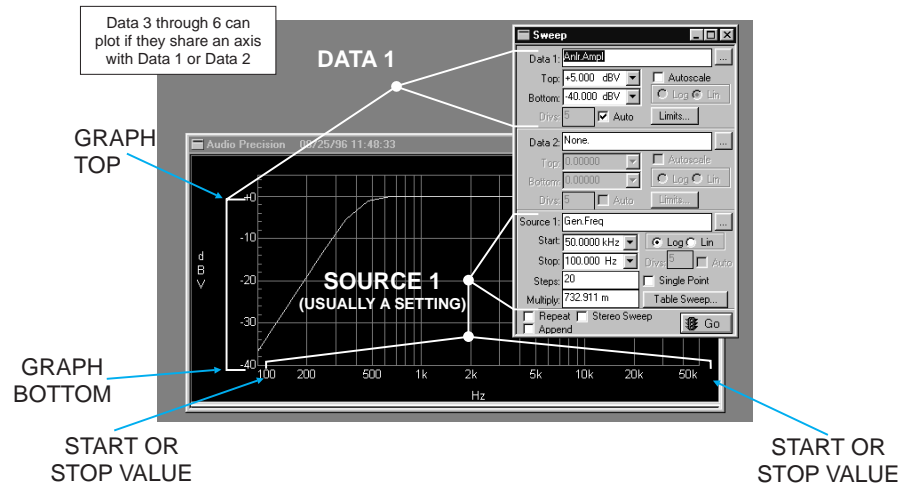
There are several reasons for obtaining test results as a sweep rather than making simple, impromptu measurements:

- **Sweep in order to display relationships among a set of measurements.** For example, frequency response is more easily seen on a graph than by attempting to remember and mentally relate a series of individual, “spot” level measurements at different frequencies. For more information, see the “Sweeps and Sweep Settling” chapter of the User’s Manual.



Exercise:

Select the test named **SWEEP1.AT2** and click OK or press Enter. Adjust the volume control if necessary for a comfortable loudness level. Click on the GO icon or press the F9 function key to run the test. Note that the readings are plotted point-by-point on the graph and listed in tabular fashion in the Data Editor as the generator frequency steps across the audio band. Note that “Gen Freq” has been selected as Source 1 at the bottom of the Sweep panel, so the generator frequency is swept as the independent variable of the test. Observe the Source 1 settings and compare them to the X axis units (Hz), log/lin graph calibration, start and stop values, and direction of sweep (Start high, Stop low). These graph and sweep parameters are all controlled by the Source 1 settings of the Sweep panel. In the Data Editor, note that there are 21 rows of data corresponding to 21 measurements. The sweep starts at the Source 1 Start value and then makes 20 steps, as defined by the Steps parameter, ending at the Stop value. Note that column 1 in the Data Editor corresponds to Source 1 and the graph horizontal axis. “Anlr Ampl” (Analyzer Amplitude) is selected as Data 1 at the top of the Sweep panel, so readings from the Analyzer Amplitude meter are plotted versus the generator frequency. Observe that the Data 1 settings define the Y axis units and graph top & bottom values. Note that column 2 in the Data Editor corresponds to Data 1 and the left-hand Y axis.



- **Sweep in order to save measurement results to computer disk.** An impromptu measurement must be manually written down to save the data. The results of a sweep test (including single-point “sweeps” for “spot” measurements) may be saved to disk as a test file or data file. For more information, see the “File Save As” section of the “APWIN Menus” chapter of the User’s Manual.



Exercise:

Use the File Save As Test command. In the File Name field of the Save As dialog, type in a file name such as the first several characters of your name and click OK. The test setup with the data just measured has now been saved under the name you supplied. Open Test **BP-SHAPE.AT2** and note that this is a completely different setup and display. Note that the Graph title bar displays the date and time at which BP-SHAPE was last run and saved. Now, Open Test with the name you supplied when you saved the test a moment ago. Press F7 which re-plots the saved data. Note that all settings, the curve, and the tabular data are exactly as when saved and that the date and time on title bar correspond to when you pressed F9 a few moments ago.

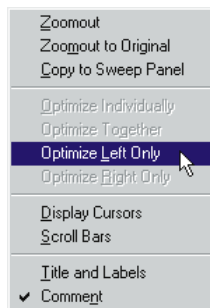
- **Sweep in order to perform additional computations on the measurement results.** For example, the measured left

channel response may be subtracted from the right channel response of a stereo device with the COMPUTE DELTA function to provide a graph of stereo balance. For more information, see the Compute chapter of the User's Manual.



Exercise:

Open Test **LINEARIT .AT2** and press F9 or click GO. This test sweeps the generator channel A amplitude downwards from -60 dBu to -110 dBu in two dB steps, measuring the level with the analyzer Level A meter. A straight diagonal trace is plotted where the generator and analyzer are perfectly linear, with some flattening visible at the low amplitude (lower left) end as the wideband Level meter becomes noise-limited. At the end of the sweep, the COMPUTE LINEARITY command is automatically invoked by this test. COMPUTE LINEARITY fits the best straight line to a specified section of the data (in this case, the data between -60 dBu and -80 dBu), then subtracts every data point from that best straight line. The result is a plot of deviation from perfect linearity. This should be essentially zero over the higher-amplitude portion of the data where noise was not a factor, so the computed data results are off the top of this graph. Click the right mouse button while the mouse cursor is on the graph. On the dialog which appears, click the “Optimize Left Only” selection.



The Optimize function automatically chooses new graph top and bottom values to best display the existing data. This new curve will show near-perfect linearity down to about -90 dBu, with increasing error in the positive direction below that since the Level meter is measuring the combination of its internal

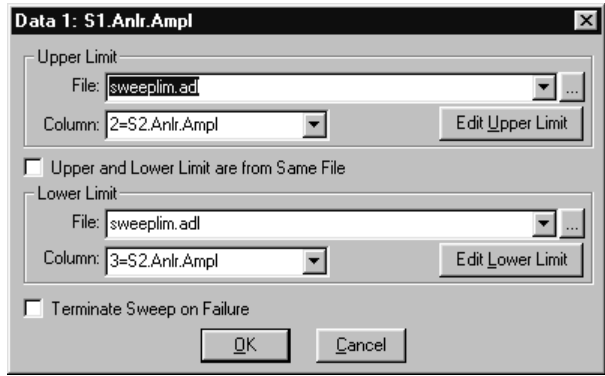
noise and a signal whose amplitude is fading into the noise level. To see how the Compute Linearity command was set up, click on the Compute menu and Linearity command. Note that the Data 1 checkbox is checked as the data to be computed and that -60 dBu and -80 dBu are entered as Start and Stop values. This tells APWIN to make the best straight-line fit to the data points between -60 and -80 dBu. The “Apply After Sweep” box is checked, which means the Compute Linearity computation will be automatically invoked following the last data point. Click Close to close the Compute dialog.

- **Sweep in order to compare measurement results to acceptance limits for GO/NO GO testing.** Manufacturing production test and regular maintenance programs typically require the comparison of measurement results against standards established by management. Rather than depending upon an operator to make those comparisons, they can be done more accurately, more rapidly, and more reliably by the computer. The computer may then take a variety of actions if data is outside limits. Actions can include halting the test, producing a log file with “passes” and failures indicated, displaying a failure message to the operator, and branching to another section of a program which could prompt the operator to make adjustments.



Exercise:

Open Test **LIMSWEEEP .AT2** . This test is similar to **SWEEEP1 .AT2** used earlier, but with upper and lower limits which plot on the graph. Press F9 or click GO. Data will plot and should be between the limits. These limits come from a limit file named **SWEEPLIM .ADL** . Click on the “Limits” button in the Data 1 section of the Sweep panel. The dialog displayed shows that **SWEEPLIM .ADL** is selected as both the Upper Limit and Lower Limit file.



Column 2 of **SWEEPLIM.ADL** is selected as the upper limit file and column 3 of the same file is the lower limit. To view the actual limit data, click either the “Edit Upper Limit” or “Edit Lower Limit” button. The limit file will be displayed identically to a Data Editor display, with asterisks in the column heading of the selected column.

Gen.Freq	**Data 1 upperlimit	Data 1 lowerlimit
0 50.0000 kHz	-30.000 dBV	-35.962 dBV
1 44.1500 kHz	-24.000 dBV	-30.629 dBV
2 39.0000 kHz	-18.000 dBV	-25.319 dBV
3 34.4500 kHz	-14.000 dBV U	-20.080 dBV
4 30.4250 kHz	-9.000 dBV U	-14.935 dBV
5 26.8500 kHz	-5.000 dBV U	-10.045 dBV
6 23.7250 kHz	-1.000 dBV U	-6.036 dBV
7 20.9500 kHz	+0.579 dBV U	-3.421 dBV
8 18.4975 kHz	+1.730 dBV U	-2.270 dBV
9 16.3375 kHz	+2.081 dBV	-1.919 dBV

Note the upper limit values in column 2 and lower limit values in column 3. When necessary, limit values can be edited from this display and the edited results saved, replacing the earlier version. Kill this tabular display. Change the Analog Generator Channel A Amplitude value from 1.000 to 1.500 Volts and run the test again. Note that the tabular display of data in the Data Editor now shows the letter “U” (for upper) at each out-of-limits value. (It may be necessary to drag the Data Editor window to a larger width to see the data). Change the generator amplitude to 0.5 Volts and run the test again. The Data Editor now shows the letter “L” (for lower) at each failure.

For further information on creating limit files, see the “Creating Limits” section of the “Limits, Data Editor, and Attached File Editor”

chapter of the User's Manual. For further information on automatically creating PASS/FAIL messages and logging errors into a Log file, see the "Log Files" and following sections under "Utilities Configuration" in the "APWIN Menus" chapter of the User's Manual.

Time Sweeps (Chart Recorder Mode)

The horizontal axis of most sweeps is an instrument setting, typically generator frequency or generator amplitude. APWIN can also make chart recorder style sweeps where one or more parameters are measured and plotted versus time on the horizontal axis. Time sweeps are often used to show wow and flutter versus time, drift in signal levels versus time, phase variations versus time, and similar applications.




Exercise:

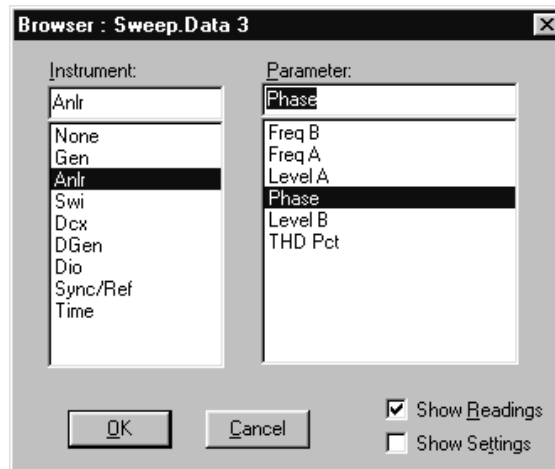
Open Test **TIME_SWP.AT2** and press F9 or GO. This test is a measurement of the analog Analyzer residual noise (22 kHz bandwidth) for five seconds. Note that External Sweep Time is selected as the Source 1 parameter, rather than the more customary case of a generator parameter setting.

Multiple Traces

As many as six different readings may be plotted onto a graph during one sweep, as determined by the Data 1 through Data 6 sections of the Sweep Panel. The Sweep Panel must be displayed at its large size in order to make settings at Data 3 through Data 6. Data 3 through Data 6 parameters can plot only if their "data domain" is consistent with one of the existing selections at either Data 1 or Data 2. For example if amplitude/level data is plotted at Data 1 and relative (%) data is at Data 2, phase data (degrees) at any of the Data 3 through Data 6 selections will not graph since degree units cannot logically plot onto either vertical axis. With Data Editor (tabular) display, up to six parameters of any type may be plotted since there is no axis calibration issue. For more information see the "Data 3 Through Data 6" section of the "Sweeps and Sweep Settling" chapter of the User's Manual.

**Exercise:**

Open Test **MULTIPLE.AT2** and press F9 or click GO to start the sweep. You should see two traces on the graph and two columns of tabular measurements: Level A in dBV (Data 1, left axis) and THD+N in % (Data 2, right axis). At the conclusion of the sweep, change the Sweep Panel to large size in order to add more measurements to the test. At the end of the Data 3 line, click on the ellipsis  button to display the Browser for selection of instruments and parameters to be plotted. In the “Instrument” column, select “Anlr” (Analog Analyzer). In the “Parameter” column, select Phase.



Go through a similar process at Data 4 to select Anlr as Instrument and Level B as parameter. At Data 5, select Anlr Freq A. Press F9 or GO. All five measurements will now appear in the Data Editor as columns 2–6. Of the three newly-added measurements, only Level B plots onto the graph, by using the Data 1 (Level A) calibration axis at the left since these are compatible domains (both amplitude). Phase in degrees and frequency in Hz cannot plot onto either an amplitude (dBV) or % axis so are not graphed. Note that the Analog Generator Channel B Amplitude was deliberately set to a different value from Channel A so that the plotted traces will be separated on the graph.

Repeating Sweeps

If the Repeat checkbox at the bottom of the Sweep panel is checked before F9 is pressed or GO is clicked, the sweep will be repeated over and over again until finally stopped manually by clicking Stop or pressing the Esc key. For further information, see the “Single vs Repeated Sweeps” section of the “Sweeps and Sweep Settling” chapter of the User’s Manual.



Exercise:

Open Test **REPEAT . AT2**. Press F9 or click GO. Note that since the Repeat checkbox is checked, the sweep repeats indefinitely until you either press the Esc key, click the Stop button on the Sweep panel, or click the red “traffic signal” icon on the Toolbar.

Appended Sweeps

When the Append box at the bottom of the Sweep panel is not checked and F9 is pressed or GO clicked, any previous measurement data from the last use of the present setup is erased and replaced with new data. If the Append box is checked, the old data is retained and new data appended. The new trace will be plotted in a different color from the older data, cycling through the six available colors (not including gray).

Both Repeat and Append may be checked, in which case sweeps will be repeated indefinitely, keeping all previous data, and using a different color for each trace. For further information, see the “Appended Sweeps” section of the “Sweeps and Sweep Settling” chapter of the User’s Manual.



Exercise:

Open Test **APPEND . AT2**. Press F9 or GO and watch the trace plot. This is a measurement of analyzer input noise versus time, at coarser time resolution than the TIME_SWPAT2 used earlier. Press F9 or GO again and note the old trace is retained since the Append checkbox is checked, and a new trace plots in another color. Check the Repeat checkbox and press F9 or GO again. Observe the automatic repetition and build-up of

multiple traces now that both Repeat and Append are checked. Press Esc or Stop to stop the sweeps.

Stereo Sweeps

When a stereo or two-channel device is to be tested, setup can be simplified by use of the Stereo Sweep checkbox at the bottom of the sweep panel. Set up the desired independent variable at Source 1 and the desired left channel measurement at Data 1, including the units, log/lin selection, graph top and bottom values, and limit file if used. Then, check the Stereo Sweep checkbox. APWIN software will automatically determine which right channel measurement parameter corresponds to the chosen left channel and will make this selection at Data 3, which causes it to share the units, graph top and bottom, etc. with Data 1. If both chosen measurements can be made simultaneously, APWIN makes a single sweep of the Source 1 parameter when GO is clicked, plotting both measurements. If a parameter was chosen at Data 1 which can only be measured on one channel at a time (THD+N, for example), APWIN first sweeps while plotting left channel data as Data 1, then automatically switches analyzer input channels (and generator output channels if necessary) and performs a second sweep plotting right channel data as Data 3. For further information, see the “Stereo Sweeps” section of the “Sweeps and Sweep Settling” chapter of the User’s Manual.



Exercise:

Open Test **MON-STER.AT2**. Adjust the volume control for a comfortable listening level. This test is set up as a monaural (single channel) Analog Generator frequency sweep measuring analog Analyzer Level A as Data 1. Press F9 or click GO and note that only a single trace (left channel only) is plotted. Check the Stereo Sweep checkbox and note that “>>Data 3>>” instantly appears in the Sweep Panel title bar to indicate that Data 3 is now in use. If you wish, expand the Sweep panel to full size to see that Data 3 now is selected to Anlr.Level B. Press F9 or GO. A single sweep is again made with two traces plotted. Both Level A and Level B plot simultaneously against the Data 1 (left vertical) axis, since the Level A and Level B meters are fully independent. Un-check the Stereo Sweep

checkbox. Change the Data 1 selection from Level A to THD+N Ratio by clicking on the ellipsis button and making the THD+N Ratio selection in the “Parameter” column of the browser. You may double-click “THD+N Ratio” or single-click and then click the OK button to close the dialog. Select Log rather than Lin at Data 1 if necessary. Press F9 and note that a single sweep is made, plotting left channel distortion only as a single plot. THD+N is measured by the Reading meter, which can measure only one input channel at any moment. Check the Stereo Sweep checkbox and note that “>>Data 3>>” appears in the Sweep Panel title bar. Press F9 and note that left channel distortion is plotted on the first sweep, followed automatically by right channel distortion plotted on the second sweep.

Table Sweeps

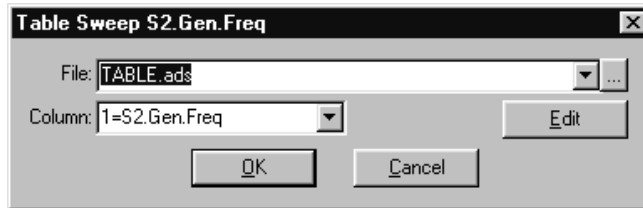
Most sweep tests allow APWIN software to determine the exact Source 1 value at all intermediate steps between the Start and Stop values. The software does this by dividing the span between Start and Stop into the specified number of steps—equal percentage steps for Log sweeps, equal value steps for Lin sweeps. When it is desired to test at specific values of Source 1, a Table Sweep can be used. A Table Sweep uses a file (.ADS file type) which contains a list of the exact Source 1 values to be used. The Source 1 Start, Stop, and Log/Lin selections on the Sweep panel then only define the graph horizontal axis, while the values in the Sweep Table define the actual Source 1 setting values. For further information, see the “Table Sweeps” section of the “Sweeps and Sweep Settling” chapter of the User’s Manual.



Exercise:

Open Test **TABLE . AT2** and press F9 or GO. Note that the data starts outside the graph area at the right end and stops short of the left axis, since the actual frequency values from the table are independent of the Sweep panel Start, Stop, and Log/Lin choices. Note that the Steps and Multiply or Stepsize fields normally displayed in the Source 1 area of the Sweep panel are not there, since the sweep table controls the Source 1 variable. Note in the Data Editor that all Source 1 values are

round numbers and are not in either a linear or logarithmic progression across the range. Click the Table Sweep button in the Source 1 area of the Sweep panel to view the Browser that was used to connect the Sweep file (table) to the test file.



The Sweep file name is **TABLE.ADS**, and column 1 within that file is used as sweep data. Click the Edit button in this dialog to display the Sweep file contents in the Attached File Editor.

**Source 1 Table	Anlr.Ampl
100.000 kHz	+19.286 dBV
40.0000 kHz	+22.814 dBV
35.0000 kHz	+22.444 dBV
30.0000 kHz	+18.220 dBV
25.0000 kHz	+20.568 dBV
20.0000 kHz	+2.223 dBV
18.0000 kHz	+12.733 dBV
1.00000 kHz	+13.288 dBV
500.000 Hz	+24.786 dBV
450.000 Hz	+21.870 dBV
400.000 Hz	+16.528 dBV
350.000 Hz	+24.102 dBV
300.000 Hz	-1.097 dBV
250.000 Hz	+19.315 dBV
200.000 Hz	+22.754 dBV

Note that the exact column 1 frequency values from the Sweep file were used as Source 1 values and thus correspond exactly with the Data Editor display of test results. The second column of the Sweep file is not used.

Sweep Speed

For the fastest sweep speeds, all measurements not being plotted should be ignored during a sweep. For some applications, it may be desirable to observe other readings while a sweep is in progress and

the consequent slow-down may be acceptable. You may select between these two modes by the Utilities Configuration menu command. On the Utilities menu, in the Configuration dialog, check “Keep all readings active during sweeps” if you wish all panel readings and bar graphs to be updated during a sweep. If the box is not checked, only the readings selected for plotting at Data 1 through Data 6 will be updated during a sweep.

Nested Sweeps

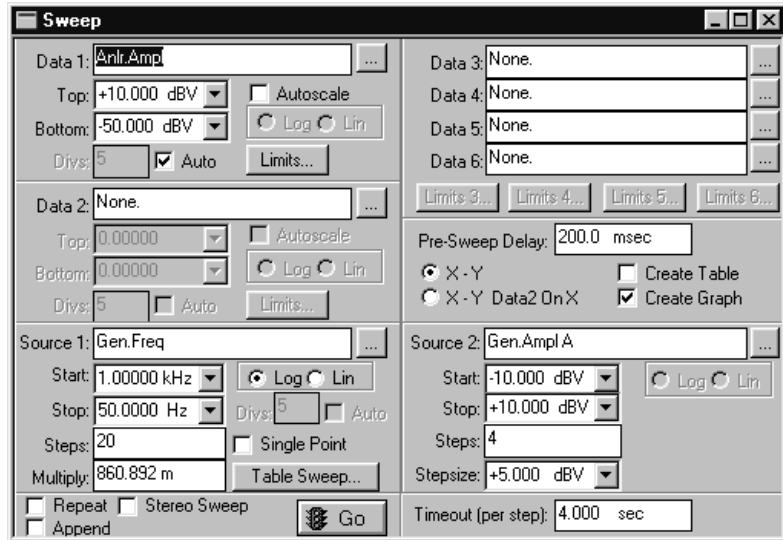
Nested sweeps are multiple sets of measurements where two different settings parameters are varied while measurements are made. One parameter is controlled at the Source 1 section of the Sweep panel and another at Source 2. At the user-specified Start value of the Source 2 parameter, Source 1 is swept through its specified Start-Stop range. Source 2 is then changed by one step from Start toward Stop and Source 1 is swept again. This process continues through the entire specified range of the Source 2 parameter, resulting in a family of curves with each curve corresponding to a specific Source 2 value. A typical nested sweep is with generator frequency at Source 1 and generator amplitude at Source 2, to show frequency response at various operating levels of a device. Nested sweeps with generator frequency at Source 1 and switcher channel at Source 2 provide measurements of many different devices or different channels of a multichannel device on the same graph. For further information, see the “Source 2 and Nested Sweeps” section of the “Sweeps and Sweep Settling” chapter of the User’s Manual.



Exercise:

Open Test **NESTED . AT2**. Press F9 or GO. Note that “>>Source 2>>” is displayed in the Sweep panel title bar, indicating that Source 2 is in use. Observe that a sweep is made across the Source 1 range with data plotted as the sweep progresses. The Graph Legend (text display under the graph) shows a Source 2 column, indicating that the Generator Channel A Amplitude is set to -10 dBV on this first sweep. Immediately after the last point of this first sweep, another sweep starts with the Generator Channel A Amplitude now at -5 dBV. In all, five sweeps are made at five values of generator

amplitude from -10 dBV to +10 dBV, each trace being plotted in a different color. Expand the Sweep panel to its large size. Note the settings at the Source 2 area in the lower right corner which produced the nested sweep—Generator A Channel Amplitude as the Source 2 parameter, four steps of +5 dB following the initial Start value of -10 dBV.



If graphic cursors (discussed below) are attached to any of these traces, a numeric display box at the top of the graph will display the Source 2 value of the trace.

External Sweeps

External sweeps make audio measurements when the test signal is not under control of APWIN software. Examples include playback of recorded test signals from test tapes or test CDs, or reception of a distant-origination sweep in a broadcasting or telecommunications application. External sweeps are set up by selecting a reading (measurement) at Source 1 instead of a setting. Most external sweeps are versus frequency, so the analyzer frequency measurement is typically the Source 1 parameter. From the frequency measurement at each step of an external sweep, the software knows where to plot the data horizontally and where to steer the bandpass or bandreject frequency if the tunable filter is used.

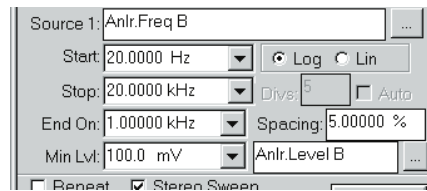
Test tapes or discs often start with a mid-band reference frequency (typically 1 kHz). This is typically followed by a series of fixed tones either progressing upwards from about 20 Hz to a final frequency of about 20 kHz, or a downwards from 20 kHz to 20 Hz. Often, the last sweep point is followed by a final mid-band reference tone. APWIN software must be told whether the sweep direction will be increasing (from 20 Hz to 20 kHz, for example) or decreasing (20 kHz to 20 Hz). Using that information, APWIN does not plot lines for the initial change from mid-band to sweep start point or the final transition from sweep final point back to mid-band, but plots only the step-to-step measurements for the sweep. To tell APWIN the expected direction of sweep, the Start and Stop values entered at Source 1 must correspond to the direction of change of the incoming signal. Voice announcements between test tones, common on analog test tapes, will be ignored by APWIN's normal data settling algorithm which disregards unstable readings and plots data only when stabilized. Noise occurring during the "dead time" between test signals may also be ignored by setting the Minimum Level parameter of Source 1 (visible only when a reading parameter is selected) to a value higher than the noise level but lower than the expected signal level. For further information, see the "External Sweeps" section of the "Sweeps and Sweep Settling" chapter of the User's Manual.

**Exercise:**

Open Test **EXTERNAL . AT2**. Press the F7 (not F9) key, which re-graphs the data stored with the test, since it is not possible to run an External sweep unless you have an appropriate player with a recorded source of tones. This data was the frequency response of a CD player measured from tracks 46-55 of the Denon Audio Technical CD 38C39-7147. Observe that the Sweep Source 1 parameter is the analog Analyzer Channel A Frequency measurement—a reading, not a setting as is customary at Source 1. Click the ellipsis button at the right end of the Source 1 line to display the Browser. Note that it is necessary to check the "Show Readings" checkbox in the lower right corner of this Browser in order to display the available readings. The normal first view of the Source 1 Browser has the "Show Settings" box checked since most sweeps are made by varying settings at Source 1.



Note that the Min Lvl (minimum level) line near the bottom of the Source 1 area shows the Analyzer Channel A Level meter as the source of minimum level information and that 100 mV has been entered as the minimum level value.



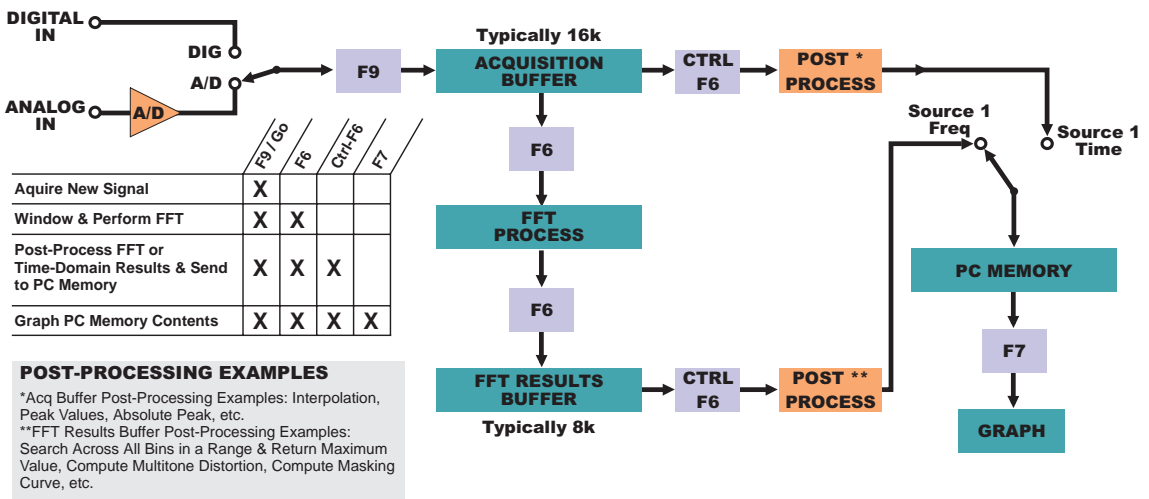
If the measured value from the Level A meter goes below 100 mV while the CD player goes from the end of one track to the beginning of the next track, no measurements will be made.

DSP Batch-Mode Operation with Time or Frequency Domain Measurements

With the DSP module present (SYS-2222 or SYS-2322 series), additional measurement modes are available including spectrum analysis and waveform display by use of the FFT spectrum analyzer (fft) or Multitone audio analyzer (fasttest) analyzer functions of the Digital Analyzer panel. These functions operate in batch mode instead of being a succession of real-time measurements as described above. For real-time sweeps, measurements are made one at a time by System Two hardware and individually sent to APWIN software in the computer for plotting and storage. The set of data points accumulates only in the computer, not in System Two. The DSP batch-mode measurements described in this section are in contrast. The entire set of measurements resides only in the DSP module of System Two, with some smaller sub-set or processed representation of the results sent to the computer for display and storage. While the end result may still be an X-Y graph with amplitude plotted on the Y axis and either frequency or time on the X axis, the process is quite different.

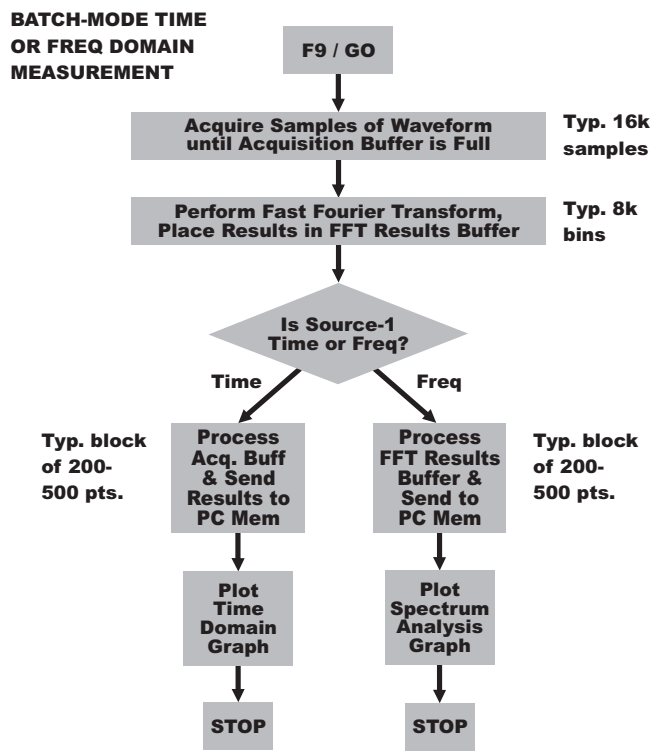
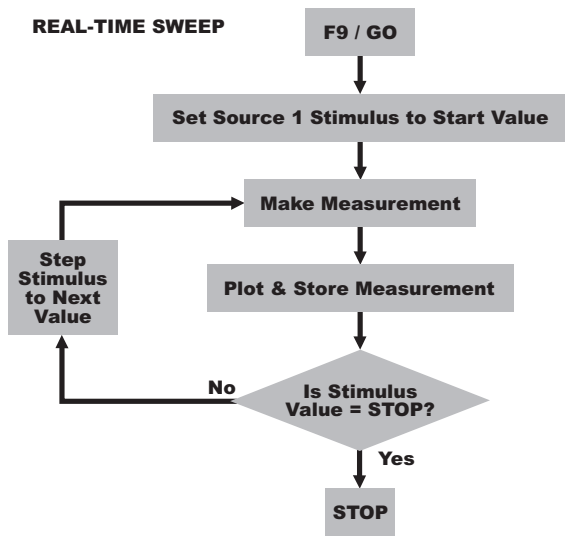
Batch mode analysis programs in System Two operate in three distinct steps:

1. acquire a sampled signal into a DSP module memory buffer for a period of time, in the form of a series of digital numbers each representing the instantaneous amplitude of the signal at the time of sampling. A typical sample rate is 48 kHz to provide a signal bandwidth somewhat above 20 kHz, and typical acquisition durations (memory buffer lengths) are hundreds of milliseconds to provide low-frequency resolution better than 10 Hz.
2. perform mathematical processing, typically a Fast Fourier Transform (FFT) plus some post-FFT processing
3. send a representative version of the results, either as an array of amplitude values versus frequency (FFT spectrum analysis) or amplitude values versus time (waveform display), to the computer for display. Typically only a reduced or simplified version of the total data in the DSP module buffers is sent to the computer since the computer screen resolution is insufficient to display the full detail present in the DSP module. Processing takes place in the DSP module to assure that this simpler version is a reasonable representation of the complete signal stored in the DSP module.



The originally-acquired sampled signal remains in DSP memory until either a new signal acquisition replaces it, the Digital Analyzer

program is changed, or System Two power is turned off. Thus, the signal may be re-processed according to different parameters and re-displayed in order to alternately look at both waveform (time domain) and spectrum analysis (frequency domain) representations, perform FFTs of different time segments of the signal, change among different post-processing techniques, use different windowing functions prior to the FFT process, etc. The complete acquired signal can be transferred to a computer disk file for storage and later down-loaded to System Two for further analysis. For further information, see the FFT-Based DSP Programs, Spectrum Analyzer, and Multitone Audio Analyzer chapters of the User's Manual.



**Exercise:**

You must have a DSP-based System Two (SYS-2222 or SYS-2322) in order to go through this example. Open Test **TIME-DOM.AT2** and press F9 or GO. If the volume control is not already adjusted for audible monitoring, adjust it for a comfortable level. You should see an oscilloscope-type presentation of five cycles of a 1 kHz sinewave on each channel. Oscilloscope (time domain) displays are produced by the FFT Time selection at Source 1. In this example, the Data 1 and Data 2 graph top and bottom have been selected to offset the two channel signals for a typical dual-trace oscilloscope view by choosing +2 and -6 Volts for Data 1 and +6 and -2 Volts for Data 2. Press the F12 function key which turns off the generator output; the loudspeaker should now be quiet. About 341 milliseconds of sampled signal are still stored in DSP memory. You may continue to observe any other portion of that signal by entering new values at Source 1 Start and Stop and pressing the F6 or Ctrl-F6 keys to send new data from the DSP module to the computer. For example, enter 337 milliseconds at Source 1 Start and 342 milliseconds at Source 1 Stop and press F6. You should now see the last several cycles in DSP memory, including the end of the acquired signal. Note that F6 and Ctrl-F6 do not cause a new signal acquisition; in fact, the generator is now off. F6 and Ctrl-F6 cause data to be transmitted from the DSP to the computer again, in accordance with the present settings of the various APWIN software panels.

To perform an FFT spectrum analysis of this same stored signal, Open Test **FREQ-DOM.AT2**. This test has FFT Freq selected at Source 1 and thus will produce a spectrum analysis (frequency domain) display (amplitude vs frequency). This test uses the same DSP analyzer program, so loading the test does not disturb the signal already present in DSP memory. This test is stored with the generator off to show that no new signal is acquired; the spectrum analysis is already stored in DSP memory. Press the F6 or Ctrl-F6 key to send data from the DSP to the computer in accordance with these new settings. The vertical axis units have been changed from Volts to dBV to best display the wide dynamic range of a spectrum analysis. The

display should show a “spike” at 1 kHz reaching to zero dBV (one Volt), a general noise baseline at about -135 to -140 dBV, and typically several harmonics of the 1 kHz signal with amplitudes around -120 to -130 dBV. Note that these harmonics are primarily generated in System Two’s A/D converters. Analog generator distortion products are more than 130 dB below the fundamental signal.

Multitone Testing

A specific synchronous multitone technique using FFT technology is called *FASTTEST*. *FASTTEST* provides extremely rapid testing of analog or digital audio devices. For example, response, distortion, and noise can be tested on both channels of a stereo device in approximately two seconds with *FASTTEST*. A typical multitone signal consists of sinewaves evenly (log) spaced across the audio spectrum. A synchronous FFT analysis (no windowing function used, thus no signal spillage) and post processing can produce most audio measurements from a single acquisition and FFT transform.



Exercise:

You must have a DSP-based System Two (SYS-2222 or SYS-2322) in order to go through this example. Open Procedure **FASTTEST .APB** and use the Procedure Run command. In rapid succession you should see a two-channel FFT spectrum analysis of the multitone signal, the frequency response, the total distortion, and noise in the presence of signal, all overlaid on the same graph. If the volume control is not turned up, adjust it so that you can hear the multitone signal. This particular multitone signal consists of 31 equal-amplitude sinewaves centered approximately at the ISO 1/3 octave standard frequencies across the audio spectrum.

For more information on multitone testing, see the “Multitone and Synchronous FFT Concepts” section of the Multitone Audio Analyzer chapter of the User’s manual.

Procedures (“Macros” or “Scripts”) Comprised of Sequences of Operations

APWIN Procedures are computer programs which control sequences of operations including opening and running tests, saving test results to disk files, comparing test results to limits and taking action depending upon whether the test is passed or failed, directly changing any settings of any panels, performing Compute functions, and all other System Two instrument control and measurement. These Procedures (similar to programs sometimes called “macros” or “scripts” in other computer documentation) also may include a full range of mathematical computations, string handling, and program flow control such as For/Next and Do While loops.

A common application for a Procedure is to perform a complete suite of tests for a particular type of device under test, such as at a final functional test station at the end of a manufacturing line.



Exercise:

Use the File Open Procedure command to open **C:\APWIN\S2\PROCEDURE\SYSCHECK\S2-22CK.APB**. Run the procedure by clicking on the Run Procedure icon if the Procedure Toolbar is displayed, or by the menu command Procedure Run. Follow the instructions of the prompting messages. This procedure will run through a series of tests which measure the key performance parameters of the analog generator and analyzer, comparing all measurements to published specifications. The procedure ends by displaying the Log file which summarizes the tests which were run, whether they passed or failed, and lists specific out-of-spec points if there are failures.

Procedures may be created in Learn Mode, accessible by the Procedure Learn Mode menu command or the start Learn Mode icon on the Learn Mode toolbar. Learn Mode writes a line of procedure language code for each mouse click or keyboard entry made by the operator going manually through a sequence of operations. Running that procedure will then duplicate the operator’s actions. Documentation on how to write and edit Procedures may be found in

the on-line Help APWIN Basic Language and Help APWIN Basic Editor menu commands or in the APWIN Basic Programmers Manual.

Equalization Functions

Test instrument generators are normally designed to have constant output amplitude as their frequency is varied, and System Two in fact has the best flatness specification in the industry. It is also possible for the System Two analog generator, digital generator, and jitter generator to have an equalization function applied (when generating sinewaves) which controls their output amplitude as a function of generator frequency. For example, the analog generator can use a deemphasis curve while making a frequency sweep into the input of a preemphasized broadcast transmitter in order to produce essentially constant deviation (modulation percentage) at all frequencies. A phonograph equalization curve may be used while testing a phono preamplifier so that essentially flat response is obtained at the preamplifier output.

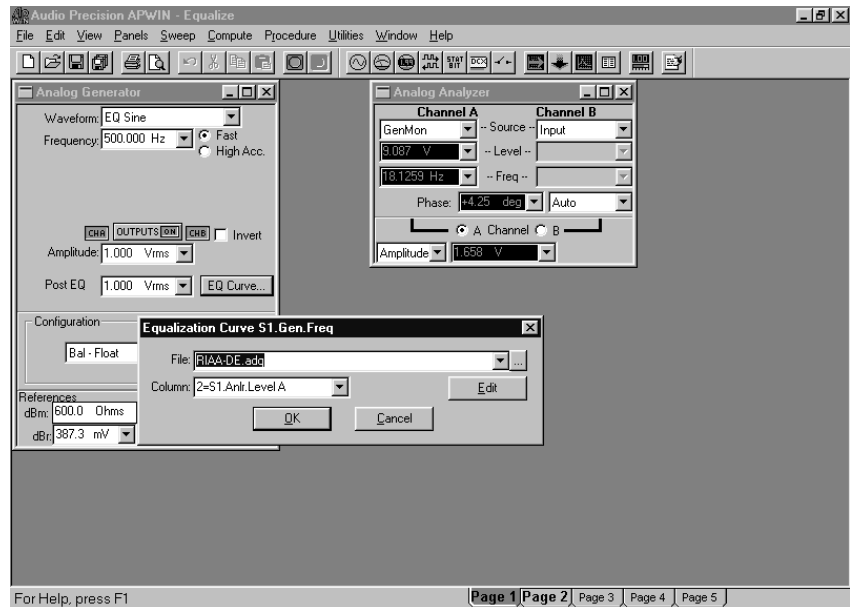
The equalization functions work by referring to a disk file which contains the desired amplitude-vs-frequency relationship. A number of standard equalization files are furnished with APWIN software and the user may also create his own. APWIN adjusts the actual (Post-EQ) amplitude of the generator, using the user-specified (Pre-EQ) amplitude and the correction factor from the EQ curve.



Exercise:

Use the File Set Working Directory command to set the current working directory to **C:\APWIN\S2\TUTORIAL\APWIN**, as it may have been changed when the example procedure was run. Open test **EQUALIZE.AT2** and press F9 or click Go. Note that the measured graph varies by about 40 dB across the audio spectrum, rather than the usual flat response within a few hundredths of a decibel. This curve shape is the industry-standard RIAA phonograph equalization curve. Go to page 1 and note that the Analog Generator has EQ Sine selected rather than normal sinewave. Note that two Post-EQ fields are now visible below the normal (pre-EQ) Amplitude fields of the generator. Try entering different values in the

generator Frequency field and note that the Post-EQ fields (and the measurements made by the analog analyzer) change at each different frequency. Pre-and-post EQ values will be equal only at 1 kHz, the frequency at which this particular equalization curve goes through zero. Click on the “EQ Curve...” button on the generator panel and note that the equalization file presently in use is named **RIAA-PRE.ADQ**. The dialog displaying the file name is used to select different equalization files.



Click on the Edit button in this dialog to see a tabular display of the equalization file contents.

A set of swept measurements can also be equalized following the end of a sweep by the Compute Equalize function. Compute Equalize uses the information in an equalization file to adjust the stored data values from every point of a sweep.



Exercise:

With the graph (page 2) of **EQUALIZE.AT2** still on screen, select Compute Equalize from the Compute menu. Note that

RIAA-DE.ADQ is selected as the Compute Equalize file. This file is the reciprocal of the file which controlled the generator amplitude during the sweep just performed. Click on the Compute and Close button and the result should now be a flat horizontal trace across the screen. Compute Equalize performed measurement equalization with the inverse of the file used during the sweep, resulting in flat overall response just as would have been accomplished when an RIAA preemphasis curve was used while making a disk recording and RIAA deemphasis used during playback of the recording.

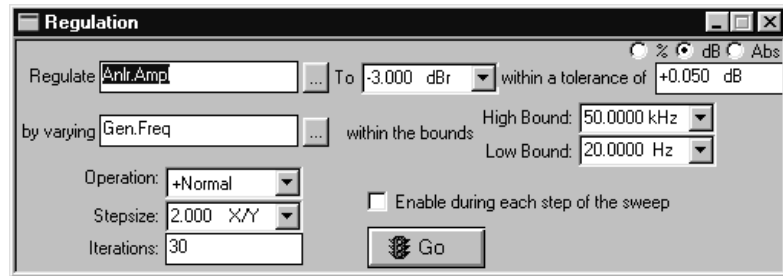
Regulation Function

Regulation is a software servomechanism which will control a setting in order to cause a reading to come to a certain target value, or to maximize or minimize. Regulation can be initiated while viewing APWIN panels and bargraphs, or can be set to operate at every Source-1 step during a sweep. Common applications of Regulation in panels and bargraph mode include varying the frequency of the generator so as to find the center (maximum response) frequency of a bandpass filter or to find the upper or lower minus 3 dB point of an amplifier or filter. Common sweep applications of Regulation are to adjust generator amplitude automatically at every step of a frequency sweep to the value that causes a specified amount of distortion (threshold of clipping) from a power amplifier, or to hold measured modulation percentage of a broadcast transmitter constant in spite of the preemphasis network built into the transmitter.



Exercise:

Open Test **REGULA-3.AT2**. This test is set up to automatically find the minus 3 dB frequency of a device under test, which in this example is the 400 Hz high-pass filter in System Two's reading meter. Adjust the volume control for a comfortable listening level. Click on the "Regulate" button and watch the Analog Generator Frequency field and the Analog Analyzer Amplitude (reading meter) field.





Bargraphs are connected to both generator Frequency and analyzer Amplitude to make it easier to watch the process. Note that the generator frequency starts downwards in large steps until the measured response is more than 3 dB down, then reverses direction and takes smaller steps until response is less than 3 dB down, reverses again, etc. Note the settings on the Regulation panel. The Regulation “engine” has been set up to regulate the Analog Analyzer Amplitude meter to -3.00 dBrA within a tolerance of +/-0.05 dB, by varying the Analog Generator Frequency within the range from 50 kHz to 20 Hz. The “Operation” selection is “+Normal”, meaning that the Regulation function is told to expect the measured amplitude to increase as the generator frequency increases. This is true on the low-frequency (high-pass) portion of a bandpass filter. Since the present measurement value when the test is loaded is above the target value of -3.00 dB, Regulation starts reducing the generator frequency in order to reduce the amplitude and eventually brings the amplitude to the target value within the specified tolerance. Change the generator frequency back to a value well above 400 Hz, change “Operation” from “+Normal” to “-Normal”, and click the “Regulate” button again. This time you will observe the frequency move higher, since “-Normal” says that amplitude will decrease as frequency increases. The regulation cycle stops when the target value is reached, having located the -3 dB frequency of the 22 kHz low-pass filter. Note that other “Operation” modes include Maximum to find a peak, Minimum to find a dip, and Linear for use with linear processes such as device output versus input.

Displaying Sweep Results

The user has considerable control over just how sweep test results are displayed. First, the user may choose graphic display, tabular display, or both by checking the Create Table, Create Graph, or both checkboxes on the large form of the Sweep panel. For more information, see the “Data Display Mode” section of the “Sweeps and Sweep Settling” chapter of the User’s Manual.

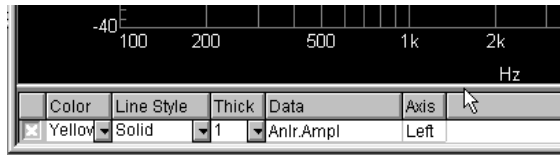


Exercise:

Open test **SWEEP1.AT2** and adjust the loudspeaker volume control to a comfortable level. Press F9 or click GO and note that measurements are displayed in both graphic and tabular formats. Expand the Sweep Panel to its large size and note that both forms were displayed because both the Create Graph and Create Table checkboxes in the right-center section of the panel are checked. Uncheck the Create Table checkbox. Kill both the graph and the data editor. Press F9 again and the graph will automatically appear to plot the data, but there will be no tabular display. Kill the graph, check the Create Table checkbox and uncheck the Create Graph checkbox. Press F9 and the Data Editor will automatically appear to display the data. Kill the Data Editor and uncheck both boxes. Press F9 and neither graph nor table will appear, but you will hear the sweep taking place. Then, click the graph icon  in the Toolbar and see that new data is already in place on the graph. Click the Data Editor icon  and see that the new data is also present in that form of display. In some manufacturing test applications where acceptance limits are used for pass/fail testing and the test operator may not be trained to interpret the data, it may be desirable to not display either graph or table in order to gain speed.

For graphic display, most control over the display is via either the Graph Legend or a right mouse button dialog. The Graph Legend is the text area below the graph, with column heads including color, thickness, line style, etc. If the Legend is not visible, use the mouse to

drag the bottom inside graph border upwards away from the outside border.



The “Data” column defines which measurement instrument is plotted into a particular trace. The “Axis” column tells whether the trace is to be interpreted against the Left (Data 1) or Right (Data 2) vertical axis calibration. Each trace may be displayed or not displayed by clicking the “X” in the corresponding row of the narrow first column of the Legend. Each trace may have any of seven colors assigned by clicking the down arrow at the right end of the “Color” column in the desired row and selecting the desired color. The width of each trace may be adjusted by selecting values from 1 to 30 in the Thickness column and row corresponding to the desired trace. If the Thickness value is 1, the Line Style column permits selecting among several line styles including solid, dash, and dot. Note that all of these selections control only the on-screen graph display. A separate set of controls on the Page Setup dialog (discussed later) control the same attributes of a printed graph. For more information, see the “Graph Legend” section of the “Graphs and Printing” chapter of the User’s Manual.



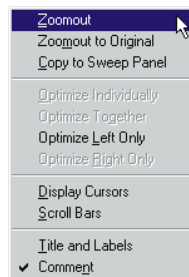
Exercise:

Open Test **MULTIPLE.AT2** and press the F7 key to plot the data already stored with this test. You should see two traces which are described in the Data column of the Legend below the graph as Anlr Level A and Anlr THD+N Ratio. You may temporarily select either of these traces by clicking in the desired row of the Data column, or de-select a trace by clicking again in the Data column. While selected, a trace will be displayed thicker than its normal setting. Click in the first column of any row to turn the corresponding trace on and off; an X is displayed in this column when the trace is displayed. Change the color of one or more traces by clicking the down arrow of the Color column in the corresponding row and selecting another color. Change the thickness to any value up to

30 units by selections in the Thickness column. Change from solid to another line style in the Line Style column, noting that only the solid line style can be displayed for Thickness greater than one.

Zooming

The user may zoom in on any section of a graph to examine the data in more detail. Zooming is particularly useful during FFT spectrum analysis, where there is considerably more resolution available in the FFT results (typically 8,192 points) than can be graphed on screen at one time. Zooming is accomplished by holding down the left mouse button while dragging a rectangular shape over the portion of graph to be expanded. It is possible to zoom in several times in succession. If either axis is logarithmic, the eventual zoom factor on that axis will be limited to a minimum span of about 20% from end to end. For a linear axis, there is no limit to the amount of zoom possible. To zoom back out, click the right mouse button while the mouse cursor is in the graph area.



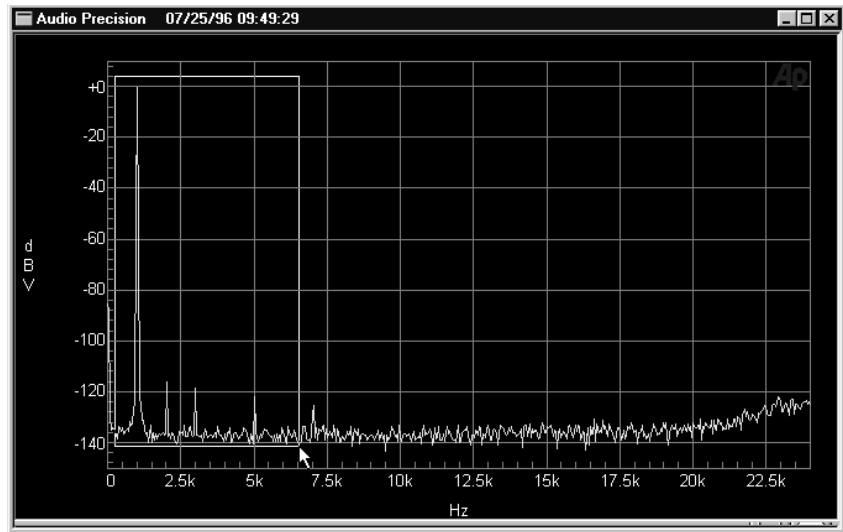
A selection of choices will be displayed as illustrated. Click on the Zoomout choice to zoom back out one level to the immediately preceding view. Additional Zoomout operations will take you back in sequence through each level of zooming. The Zoomout to Original command will take you to the original view corresponding to the Sweep panel settings. For more information, see the “Zooming” and “Right Mouse Button Features” sections of the “Graphs and Printing” chapter of the User’s Manual.



Exercise:

For DSP Units:

If your instrument is an SYS-2222 or SYS-3222, Open Test **FFT .AT2**. Press F9 or click GO to acquire, process, and display new data. You should see the fundamental 1 kHz signal as a “spike” of 0 dBV amplitude plus several low-amplitude harmonics and a noise floor across the audio spectrum. With the mouse cursor above and to the left of the 1 kHz signal, hold the left mouse button down and drag a rectangle down and to the right, to about 6 kHz and -140 dBV.



Release the mouse button. The graph will be replaced with a new zoomed-in graph of the area within the rectangle. Note that the Utilities Configuration dialog must have the “Re-process FFT on zoom” box checked to obtain the behavior described here. Zoom again and again into still smaller areas. Eventually, as the horizontal axis span is reduced to a few hundred Hz, you will begin to see the actual FFT bin structure with (at this sample rate and record length) bins 2.92 Hz wide. You may zoom back out in the same steps that you zoomed in, or zoom all the way back to the original settings in one step. To zoom out, click the right mouse button while the mouse cursor is on the graph. On the dialog which is displayed, click either on “Zoomout” to zoom back one step, or “Zoomout to Original” to zoom back to the original settings. Use the Zoomout to Original command, then zoom in to approximately the area from 500 Hz to 2.5 kHz and 0 dBV to -140 dBV. Click

the right mouse button and select “Copy to Sweep Panel”. Observe that this command copies the present graph horizontal and vertical end point values to the Source 1 Start and Stop and Data 1 and 2 Graph Top and Bottom Settings on the Sweep panel, replacing the original settings. Press F9 or GO and note that a new acquisition is made but now only the portion within these new settings is displayed. However, the entire acquired signal (0 to 24 kHz) is present in the FFT results buffer so it is possible to manually change the Sweep panel settings to a wider range and press Ctrl-F6 or F6 to see the graph.



Exercise:

For Non-DSP Units:

If your instrument is an SYS-2022, Open Test **BP-SHAPE.AT2**. Press F9 or click GO. This test fixes the analyzer bandpass filter at 1 kHz and sweeps the generator to make a graph of the filter shape. With the mouse cursor above and to the left of the 1 kHz filter peak response, hold the left mouse button down and drag a rectangle down and to the right, to perhaps 5 kHz and -50 dBV. Release the mouse button. The graph will be replaced with a new zoomed-in graph of the area within the rectangle. Zoom again and again into still smaller areas. You may zoom back out in the same steps that you zoomed in, or zoom all the way back to the original settings in one step. To zoom out, click the right mouse button while the mouse cursor is on the graph. On the dialog which is displayed, click either on “Zoomout” to zoom back one step, or “Zoomout to Original” to zoom back to the original settings. Use the Zoomout to Original command, then zoom in to approximately the area from 700 Hz to 2.0 kHz and 0 dBV to -20 dBV. Note that near the peak of the filter it can now be seen that the graph consists of straight-line vectors connecting individual measurement points which were spaced by about 60 Hz near 1 kHz (Multiplier about 0.944). Click the right mouse button and select “Copy to Sweep Panel”. Observe that this command copies the present graph horizontal and vertical end point values to the Source 1 Start and Stop and Data 1 and 2 Graph

Top and Bottom Settings on the Sweep panel, replacing the original settings. Press F9 or GO and note that a new sweep is made, still with 80 steps but now spread only over the 700 Hz-2 kHz span (Multiplier about 0.987), thus producing a smoother and more accurate rendition of the filter shape.

Optimizing

The graph Top and Bottom values entered by the user on the sweep panel are usually an estimate of the vertical range which actual data may occupy. Incorrect estimates or changing data may result in data being plotted off the top or bottom of the graph. Clicking the Optimize commands (reached by clicking the right mouse button in the graph area) following a sweep will cause an automatic selection of graph top and bottom values to best display the actual data in memory. The Optimize Left Only and Optimize Right Only commands each affect only the data plotted against the left (Data 1) or right (Data 2) axes, respectively. If both Data 1 and Data 2 are in use with compatible units, the Optimize Individually and Optimize Together choices will also be available. Optimize Individually makes independent selections of the best graph top and bottom for the left and right axis data. Optimize Together selects the same graph top and bottom values for Data 1 and Data 2 so that the traces may be directly compared. For more information, see the “Optimize Features, General” section of the “Graphs and Printing” chapter of the User’s Manual.



Exercise:

Open Test **OPTIMIZE.AT2**. Press F9 or click GO. The graph should show the Channel A Level meter plotted as an essentially horizontal straight line at 0 dBV on a ± 2 dBV scale at the left (Data 1) axis. The Bandpass function of the reading meter, with the bandpass filter fixed at 1 kHz, should be plotted on a $+2/-50$ dBV scale at the right (Data 2) axis. Click the right mouse button on the graph and select Optimize Individually. The left scale goes to a span of a few thousandths of a dB to show the extremely small variations from flatness of the level meter. The right scale expands vertically downwards to about -70 dBV to show the bandpass skirts which were previously off

scale below the original -50 dBV graph bottom. Click the right button again and select Zoomout to Original to bring back the original view. Now, use the right button again and select Optimize Together. Now, the axis spans are set to show the full range of measurement data on both traces with the left and right vertical scales both set the same to permit easy, direct comparison of absolute values.

Cursors

Either of two cursors can be set to any measurement point. The X and Y axis values of each cursor are displayed numerically in “floating” display boxes in the graph margins. The X-axis numeric display is directly below each cursor position. The Y-axis display is horizontally aligned with each cursor position, with the Data 1 numeric display on the left axis and the Data 2 display on the right. Numeric display boxes at the top of the screen show the dX and dY (difference in X and difference in Y) values between the two cursors, and the Source 2 value for the trace if a nested sweep is displayed. To enable cursors, click the right mouse button on the graph and select Display Cursors. When initially displayed, the cursors will be vertical green lines across the full height of the graph, not attached to any displayed trace.

Click in the Data column on the desired trace description in the Graph Legend below the trace to select a specific trace for the cursors to measure.

	Color	Line Style	Thick	Data	Axis	Cursor1	Cursor2
✘	Yellow	▼ Solid	▼ 1	▼ Anlr.Level A	Left
✘	Green	▼ Solid	▼ 1	▼ Anlr.Bandpass	Righ

The Graph Legend, as discussed earlier, is the text area immediately below the graph. If the Legend is not visible, use the mouse to drag the bottom inside graph border upwards away from the outside border. When a cursor is connected to a trace, the cursor will snap to the horizontally-nearest actual measurement point and jump from point to point as it is moved. Cursors may be dragged horizontally with the left mouse button, first clicking in the X-axis numeric display box immediately below the cursor position or anywhere within the graph area on an imaginary vertical line passing through the cursor position. When dragging a graph cursor, the mouse

cursor shape changes from its usual arrow to a double-headed arrow with the numeral “1” or “2” indicating which cursor is being controlled. Once selected by the mouse, cursors can also be moved with the horizontal arrow keys. Arrow key control can be shifted between the two cursors by a Shift-arrow key operation. When a trace is initially selected, both cursors are attached to the same trace. Cursors may also be connected to two different traces if both are in the same “domain”; for example, if both are level or amplitude readings. Cursors may not be connected to different traces if, for example, one is distortion percentage and the other is amplitude. To switch a single cursor to another trace, first click in the Graph Legend column heading for the cursor to be switched (Cursor1 or Cursor2). The column head will change from the Cursor name to “**Select**”.

	Color	Line Style	Thick	Data	Axis	** Select **	Cursor2
✕	Yellow	Solid	1	Anlr.Level A	Left
✕	Green	Solid	1	Anlr.Bandpass	Righ

Then, click in the row under this column head corresponding to the other trace desired. An asterisk displayed in the cell intersection of a cursor column and data trace row indicates cursor “connection”. When cursors are connected to two different traces and dragged to the same horizontal value (dX display = zero), the dY display shows the difference in the two plots at that point. For more information, see the “Display Cursors” section of the “Graphs and Printing” chapter of the User’s Manual.



Exercise:

Open Test **OPTIMIZE .AT2** and press F7 to display the stored data, or F9 or GO to make a new sweep. Click the right mouse button on the graph and select “Display Cursors”. A vertical green line should appear near the left of the graph. Click in the X-axis numerical display box at the bottom of this cursor line and drag the cursor back and forth horizontally. Note that the cursor may be set to any horizontal value; it is not snapping between measurement points because it is not yet attached to a trace. Likewise, note that there are no numeric display boxes on either vertical axis since the cursors have not been attached to either trace. In the Data column of the Legend, click in the “Anlr Bandpass” row. This selects that trace and connects both

cursors to it. The cursor shape changes to an X with a circle. Numeric display boxes appear on the vertical axis, in line horizontally with the cursor position. Since the Anlr Bandpass trace is calibrated at the right (Data 2) axis, the numeric display boxes are at the right vertical axis. Drag either cursor by clicking in the X-axis numeric display box at the bottom and note that the cursor snaps from measurement to measurement. Click in the “Anlr Level A” row of the Data column which selects that trace and note that the cursors move to the Level A trace and their numeric readout boxes are now on the left vertical axis since this is the Data 1 trace. Click in the “Cursor 1” column head and it will change to “**Select**”. In the Cursor 1 column, click in the “Anlr Bandpass” row. Now, cursor 1 should be on the Anlr Bandpass trace (numeric readout at right) while cursor 2 remains on the Anlr Level A trace with readout at the left. Drag one of the cursors until the two cursors are vertically aligned, as indicated by a value of 0.0000 Hz in the “dx” numeric readout at the top of the graph. The “dy” readout will now show the vertical axis difference between the two traces.

Graph Title and Labels

A screen-displayed or printed graph may have a user-furnished title displayed in the graph title bar in addition to (or instead of) “Audio Precision” and the date and time of the test. The left and right vertical axes (Data 1 and Data 2 calibration) and the bottom horizontal axis (Source 1 calibration) may have user-furnished text in addition to or instead of the measurement unit normally shown there. The top horizontal axis may display user-furnished text. All of these optional text displays on the graph are controlled by the Title and Labels command in the dialog displayed by clicking the right mouse button in the graph area.

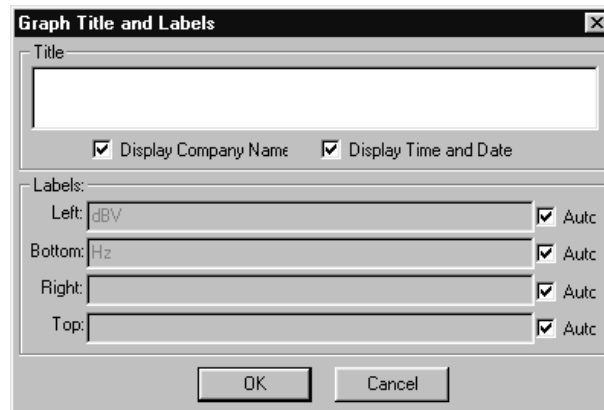


This command displays another dialog. For more information, see the “Title and Labels” section of the “Graphs and Printing” chapter of the User’s Manual.



Exercise:

Click the New Test icon. Go to Page 2 where the graph is displayed. Click the right mouse button on the graph, and in the dialog which appears, click “Title and Labels”. The Graph Title and Labels dialog will be displayed.



Type several words of text into the “Title” area. Note that this text also immediately appears in the graph title bar, pushing the date and time to the right. Un-check and re-check the “Display Date and Time” checkbox and note that the date and time disappear and re-appear. Un-check the “Auto” checkbox at the right end of the “Labels: Left:” row and type in several characters—before, after, or instead of the “V” (Volts) unit indicator. You may need to drag the Graph Title and Labels dialog to another part of the screen to note that the left axis

label displays the new characters as you type. Similarly, you may wish to change the Top, Bottom, and Right labels. Click the OK button to close the dialog.

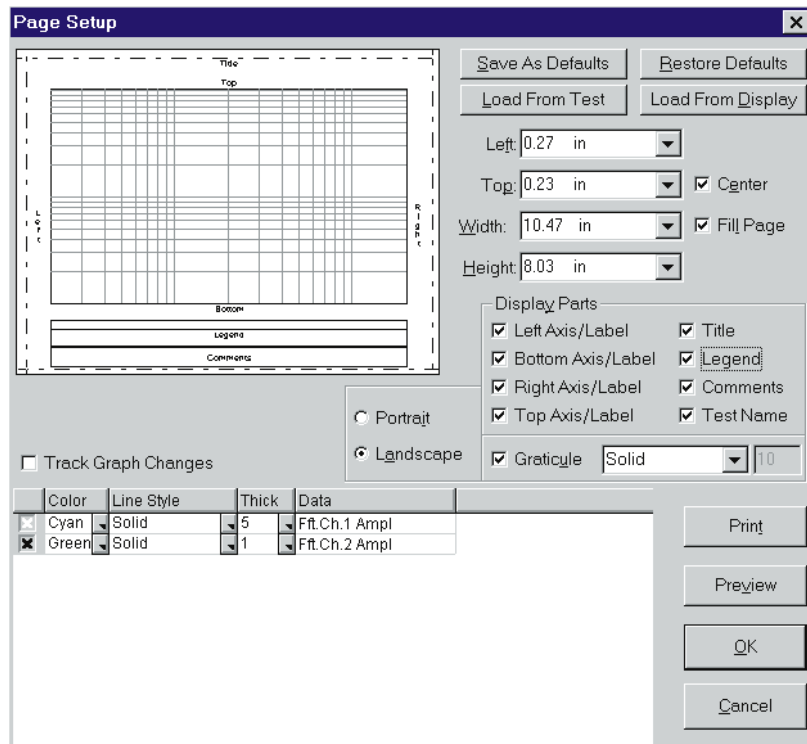
Printing Graphs

Printer and Print Driver Installation

An appropriate printer driver for the printer to be used must be installed to Windows before APWIN software can preview or print graphs. If several printer drivers have been installed on your computer, the File Print Setup command may be used to select among them.

Page Setup and Print Preview

The currently-displayed graph may be printed via any printer, monochrome or color, installed to Windows. User control of the appearance of the graph is via the File Page Setup command. Page Setup permits selecting portrait or landscape orientation of the graph on the page, control of graph size and location on the printed page, line styles (solid, dash, dot) or line thickness, color if a color printer is in use, whether or not each trace is printed, and whether or not the graph title, each individual label (left, right, top, bottom), the graph legend, and any comments in the comments area under the graph is printed.



The graph may then be previewed on screen by the File Print Preview command before actually sending it to the printer. For more information, see the “Printing Graphs” section of the “Graphs and Printing” chapter of the User’s Manual.



Exercise:

Open Test **OPTIMIZE.AT2** and press F7 to display the data stored with the test. Select File Page Setup and you should see the dialog illustrated. Select Portrait, then select Landscape to note how each print-out mode would be oriented on paper. Un-check the “Center” and “Fill Page” checkboxes at the upper right of the dialog. Now, you may use the mouse to drag the graph representation to different sizes and shapes and drag its position on the page. When the mouse cursor is positioned over an edge or corner of the graph, the cursor shape changes to a vertical, horizontal, or diagonal double-headed arrow indicating which dimension of the graph will change if dragged. When the mouse cursor is over the center section of the graph,

the shape changes to a four-headed arrow indicating that the entire graph, at its present size and shape, can be dragged to a different position on the page. As a more precise alternative to using the mouse, numbers may be typed into the “Left”, “Top”, “Width”, and “Height” boxes to set the size and location of the graph on the page. Check and un-check the various checkboxes in the “Display Parts” section of the dialog (Labels, Title, Legend, Comments) and note that each of these text elements may be independently chosen for printing. Note also that the graph area expands or shrinks when these text elements are turned on or off, in order to keep the entire graph plus text at the specified size. In the Legend under the graph representation, note that Color, Line Style, and Thickness may be set to the desired conditions for the printed graph, independently from the screen display of the graph. Click the “Preview” button on the Page Setup dialog to jump directly to the Print Preview screen. Print Preview shows a representation of how the final printed page will appear. If a color printer has been set as the default printer by the Print Setup dialog, the Preview will be in color. If a monochrome (typical of laser printers) is the default printer, the Preview will be monochrome. On the Preview dialog, the mouse cursor shape changes to a magnifying glass when over the actual page area. Clicking at any point produces a zoom-in view centered at the cursor location. A second level of zoom is available by a second click. A third click returns to the original full-page preview. It may be necessary to return to the Page Setup dialog for further changes and then to again preview the changes before printing. Printing may be accomplished by clicking the Print button of the Print Preview dialog or the Page Setup dialog, clicking on the Print icon on the Toolbar, or using the File Print menu command. For more information on printing graphs, see the “Printing Graphs” section of the “Graphs and Printing” chapter in the User’s Manual.



APWIN Simplified for System Two

Book Two



Tutorial

Getting Started with Multitone Testing

Getting Started with Procedures

Introduction To Multitone Testing

What is a multitone?

A multitone is a complex waveform containing a number of simultaneous sinewaves. It may contain only a few tones, or it may contain a few hundred.

Since most conventional signal generators can only generate one sinewave at a time, a Digital Signal Processor (or DSP) must be used to generate a multitone.

Why test using multitones?

While not useful in every situation, multitone testing can offer several advantages over conventional testing methods.

First, it is generally much faster. Using a multitone, you can test at a number of frequencies simultaneously. Conventional techniques would require stepping through each frequency at which you would like to take data, making a measurement at each step.

Second, you can take many measurements from a single short burst of a carefully selected multitone. This can drastically shorten the amount of time that your device-under-test must carry test signals. This is especially useful for on-air testing.

Third, a multitone can be designed which resembles the spectral content of your program material. While conventional measurement techniques rely on a single sinewave at a time to indicate how your device will respond to program material, multitones allow you test under much more realistic conditions.

Fourth, the analyzer can be set up to recognize the multitone and automatically begin measuring, ignoring program material.

Fifth, our implementation of synchronous multitone stimulus and acquisition provides the best known method of measuring noise in the presence of a signal.

Concepts

How Multitone Testing Works

This section briefly explains the ideas that make multitone testing work. We won't go into a lot of specific details, just enough for a solid understanding of the test methods. Following chapters will fill in the details about how exactly to set up the measurements.

As with most audio tests, we use a *stimulus-response* arrangement to characterize the performance of our device-under-test. This means that we create some sort of stimulus, have the device-under-test operate on it, and then measure the response.

For stimulus we use a multitone. It is very important that we know exactly what frequencies make up the multitone, and the amplitude of signal at each frequency. In fact, the particular measurements we desire will affect what frequencies we include in our multitone.

The multitone is generated digitally by a DSP. A DSP can generate a signal containing a large number of sinewave components, and give you a lot of control over the frequency and amplitude of each component.

Sometimes the multitone can be generated by the device itself, for output-only tests. Computer multimedia devices can generate multitones themselves, usually from a computer file that contains the multitone data.

For digital measurements, the digital multitone will be sent straight out to the device-under-test in digital format. If the device has analog inputs, however, the digital signal must first go through a digital-to-analog converter (or D/A) inside the System Two, where it will be converted to an analog signal. This analog multitone will then be sent out to the device-under-test.

After the signal has passed through the device-under-test, it must be in digital format to be analyzed. This means that if the signal is analog, it must be 'sampled' with an analog-to-digital converter (or A/D) inside the System Two to convert it to a digital signal so that it can be processed by the DSP.

The DSP then performs an FFT (or Fast Fourier Transform) on the signal. The FFT is a mathematical formula that splits the multitone (or any signal) into all of its sinewave components.

The FFT divides the frequency spectrum into a large number of 'bins'. Each bin represents a narrow range of frequencies. The value of each bin will then be the combined amplitude of all sinewave components that fall within that frequency range. For example, if we do an FFT on a single sinewave, all the bins will have very small values except the bin containing the frequency of our sinewave. This bin will have a value corresponding to the amplitude of the sinewave.

If a multitone is used for stimulus, each of the multitone's sinewave components will fall into a bin. All the other bins will contain noise, distortion, and intermodulation distortion products. Since we know the frequency content of our multitone, we can analyze the FFT in a number of ways, depending on what information we desire.

Analysis of the FFT results takes a number of forms depending on the measurement mode. The measurement mode determines what kind of post-processing the DSP will perform to analyze the FFT results. For every data point requested by APWIN, the DSP will use the measurement mode, and the FFT results, to determine a number to send back to APWIN to be graphed.

There are six measurement modes:

Spectrum: when APWIN requests data at a certain frequency, the DSP will return the value of the highest bin between the last point requested and the current point requested.

Response: when APWIN requests data at a certain frequency, the DSP will return the amplitude of the bin containing that frequency.

Distortion: when APWIN requests data at a certain frequency, the DSP will return the sum of all the bins between the last point requested and the current point requested, excluding the bins containing fundamental tones.

Noise: when APWIN requests data a certain frequency, the DSP will return the sum of every alternate bin between the previous point requested and the current point requested.

Masking: This special mode is used to generate a masking curve, which is used in codec testing to determine whether distortion components are audible.

Crosstalk: This special mode (only available for System Two) only returns the value of bins with tones present in the opposite channel.

The selection of measurement mode is made on the Digital Analyzer panel in APWIN. All multitone testing uses the *FASTTEST* Analyzer.

The following sections describe the various types of measurements that can be performed using multitone techniques. In general, when we are creating a multitone measurement, there are three main things we need to consider:

1. The multitone to generate. This will depend on what type of measurement we want to make and how many points we want in our final graph.
2. The measurement mode to use. This will depend only on the type of measurement we wish to perform. Different measurements require different post-processing of the FFT bins to extract the desired information.
3. The points to request for the graph. This will depend on the multitone we are using and the type of measurement desired. In some cases, choosing the wrong points for the graph will completely invalidate the measurement. In other cases, the points chosen may not be particularly important.

Response Measurements

Frequency response measurements are obtained using a multitone containing any number of frequency components, usually all at equal amplitudes. One tone is required for each data point that you would like in your final graph, at the frequency of that point.

After the tone has been generated or processed by the device-under-test, the signal is acquired and an FFT is performed.

APWIN then takes one point from each bin containing one of the original tones. The amplitude of the bin is graphed versus the frequency of the tone.

Here is a diagram of a response measurement:

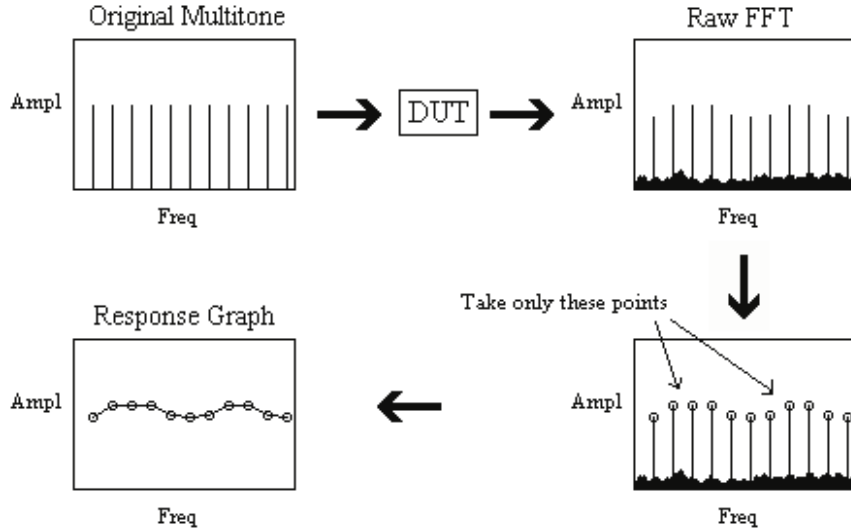


Figure 2-1 Diagram of a Response Measurement

The choice of multitone depends entirely on what points you want to graph. A common choice for this type of measurement is the ISO31 waveform, provided with your software. This contains 31 tones, spaced 1/3 octave apart.

If you want more tones, you may want to make your own multitone using MAKEWAVE (see Chapter). The phase or spacing of the tones does not matter - you can make them however you want. If you are using signal triggering, you must follow the guidelines listed in Chapter .

You do not need to use a sweep table provided you load the generator with the same multitone waveform you are measuring. The DSP will automatically detect which bins are valid for response, and interpolate points in between. However, it is ideal to use a sweep table containing the points of the tones in your multitone. If you use MAKEWAVE to generate the multitone, it can generate the data for the sweep table also. If you are using ISO31 for your multitone, you can use the sweep table "ISO31.ads" which is also provided with your software.

For a response graph, use the 'response' measurement mode.

Crosstalk Measurements

Crosstalk measurements are obtained by sending a multitone to one or more channels of a multi-channel device and then measuring the non-driven channel. Similarly to response measurements, you need one component in your multitone for each data point you want on your final graph.

System Two has a 'crosstalk' measurement mode which makes crosstalk measurements easier and also facilitates simultaneous crosstalk measurements on both channels.

To begin with, the DSP is loaded with a stereo multitone waveform. The waveform must have some tones in each channel that do not exist in the other channel. In addition, it may have tones that exist in both channels (but these tones cannot be used for crosstalk measurements). Often the crosstalk tones are provided in pairs, with one tone in one channel and one in the other, with enough difference between the two tones that they will fall into different bins. Xtlk.ags, provided with the samples, is a good choice.

When the crosstalk measurement mode is used, the DSP first performs an FFT on the original generator waveforms to identify the 'crosstalk tones' for each channel. In order to qualify as a crosstalk tone, a frequency component must exist in one channel but not the other, and must be within 40 dB of the highest-amplitude tone in the multitone.

After the acquisition and FFT, the DSP then checks the amplitude of the non-driven channel in each of the crosstalk tone bins. It then divides the spectrum into a new set of bins, with each bin centered on one of the crosstalk tones for that channel. As APWIN requests each point, the DSP returns the value of the bin containing that frequency.

Because of this rebinning process, it is not necessary to use a sweep table. However, if you don't use a sweep table, your results may be misleading, if you have only a few crosstalk tones, due to the rebinning process. The most accurate display will result from using a sweep table with points that approximate the crosstalk tones in your waveforms.

Here is a diagram of a System Two crosstalk measurement:

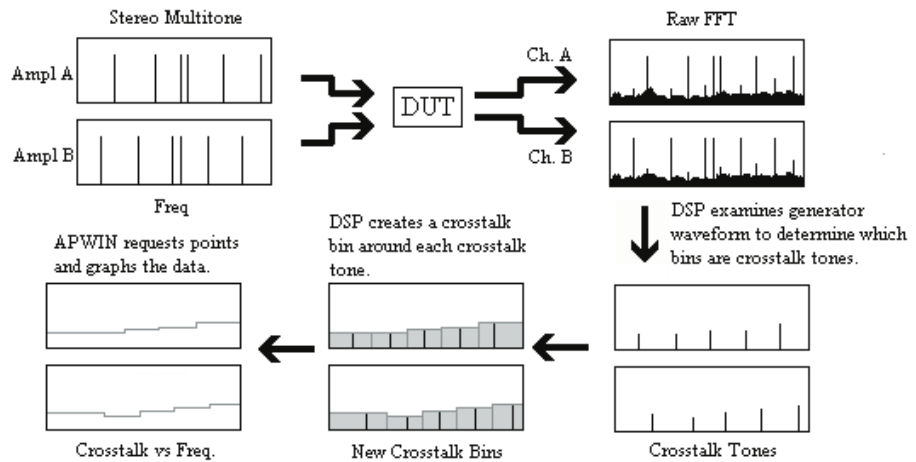


Figure 2-2 Diagram of a System Two Crosstalk Measurement

Most commonly, a multitone waveform is used that has both crosstalk and non-crosstalk tones. The crosstalk tones are placed in pairs, with one tone in each channel, spaced a few percent apart in frequency. Then a sweep table is used containing one point for each pair, approximately centered between the two points in each pair. This way, each point in the graph represents a single-point crosstalk reading at approximately that frequency.

When testing crosstalk in the presence of signal, you also need to be careful of distortion products. If channel A has a distortion product that falls into the same bin as a crosstalk tone from channel B, you will get a very high crosstalk reading from that bin. To avoid this, make certain that none of channel A's crosstalk tones is an exact multiple of any of channel B's tones, and vice-versa.

Phase Measurements

In addition to the normal amplitude bins, an FFT generates phase bins. There is one of these bins for each of the amplitude bins, covering the same frequency range.

These bins normally contain the phase of signal at that frequency, relative to the start of the measurement. This is called the 'Independent Phase' data, and it is not very useful by itself. However, we can get data about the difference in phase between two channels by subtracting the independent phase of one channel from the independent phase of the other. This will give us the interchannel phase, which is identical to a normal phase reading. For each tone present in both channels of your multitone, you can take a phase reading at that frequency. This allows you to generate an entire phase-shift-versus-frequency graph from one multitone.

In order to obtain the interchannel phase data, you must set the 'Ch.2 Phase Display' setting on the Digital Analyzer panel to 'Interchannel', and then set the Data 1 parameter on the Sweep panel to 'Fasttest.Ch.2 Phase'

Phase measurements can use ISO31, or any other collection of tones you wish to put into a multitone, with one tone for each data point. The phase tones should be sent through both channels simultaneously, and should have the same phase. You should use a sweep table containing the same points as the tones in your multitone.

Here is a diagram of a phase measurement:

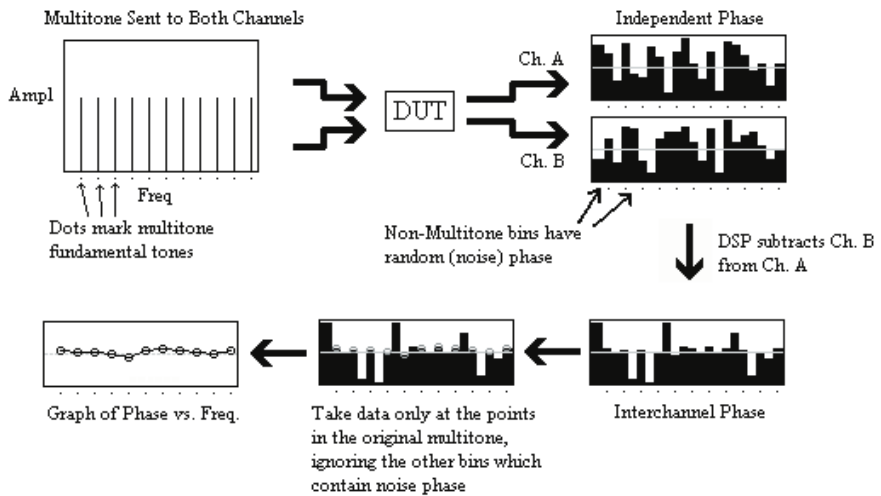


Figure 2-3 Diagram of a Phase Measurement

The measurement mode should be set to ‘Spectrum’ for phase measurements.

Distortion Measurements

Distortion measurements taken with multitone techniques are fundamentally different from conventional distortion measurements.

When we take a conventional distortion measurement, we use a single tone for stimulus. Then, for the analysis, we use a notch filter to eliminate the tone and measure the remaining signal.

The multitone equivalent generates a number of simultaneous tones, performs an FFT, and then sums up the contents of all the bins except those containing fundamental tones. These bins will contain all the distortion products, all the intermodulation distortion products, and all the products caused by the various products intermodulating with each other.

This reading is not directly comparable to a conventional distortion reading, but it is better in several ways. First, it is faster than a conventional distortion sweep. Second, this type of testing may reveal troublesome intermodulation products that a conventional distortion test may miss.

For this test we use the 'Distortion' measurement mode. This measurement mode tells the DSP to add up the bins. Each time APWIN asks for a data point, the DSP will add up all the bins between the last bin requested and the current bin requested and send the total back to APWIN for graphing.

Here is a diagram of a multitone distortion measurement:

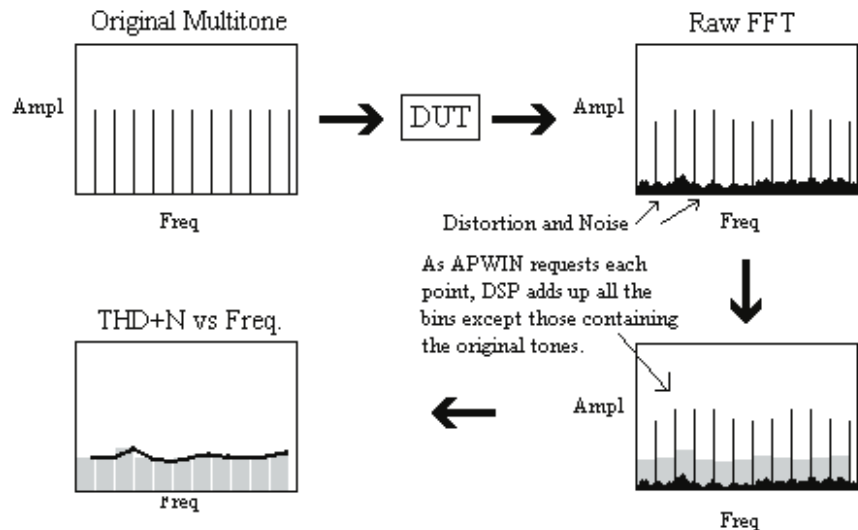


Figure 2-4 Diagram of a Distortion Measurement

The DSP will automatically skip any bins that contain fundamental tones. It does this by first analyzing the generator waveform to determine which frequencies contain fundamental tones. Therefore, you must have the generator buffer loaded with the same multitone you are analyzing, even if you are sampling a signal from another source. The generator does not have to be turned on.

The advantage of automatically excluding the bins containing fundamental tones is that a sweep table is not necessary. However, a

sweep table may be used to define the intervals over which each sum is taken. This way, the distortion measurements can be divided into ‘critical bands’ of distortion, and the distortion in each of the bands measured separately.

You can also measure specific harmonics of any of your tones, and apply limits to them. In order to do this, you need to make certain that none of the harmonics of interest fall in the same bins as other tones in your multitone. This means that you will want to avoid placing your tones on multiples of one another. If you place one tone at 1 kHz, you probably don’t want to place another one at 2 kHz or 3 kHz. 2.1 kHz, however, would be fine.

This can be a very valuable test, since most distortion problems involve either even-harmonic or odd-harmonic distortion products. In these cases, at least one of the first two harmonics is usually quite prominent. Therefore, it can be very informative to measure the first two harmonics. You probably would want to do a total distortion test (as described above) also, just to make certain you are not missing anything.

To measure specific harmonics, you would not use the Distortion measurement mode, since it would add up all the bins between each point. You would use the Response measurement mode, which will plot only the value of the requested bin. Your sweep table should contain the frequencies of all harmonics that interest you.

Noise Measurements

Multitone techniques provide excellent noise measurements that are comparable to conventional measurements, but faster. In addition, multitones provide the only known technique for measuring noise in the presence of signal.

In order to make noise measurements, you must use a Transform Length that is twice the length of the waveform file. We haven’t discussed Transform Length yet, but you can read more about it in the chapter on Advanced Concepts, Chapter 6. This setting controls the number of samples gathered by the DSP before performing the FFT.

The Transform Length affects the frequency resolution, or the size of the bins. If you double the Transform Length, you get twice as many

bins, and they are only half as wide (they only cover half as much frequency).

Similarly, the length of the waveform file controls the generator frequency resolution, or the distance between each possible frequency it can create.

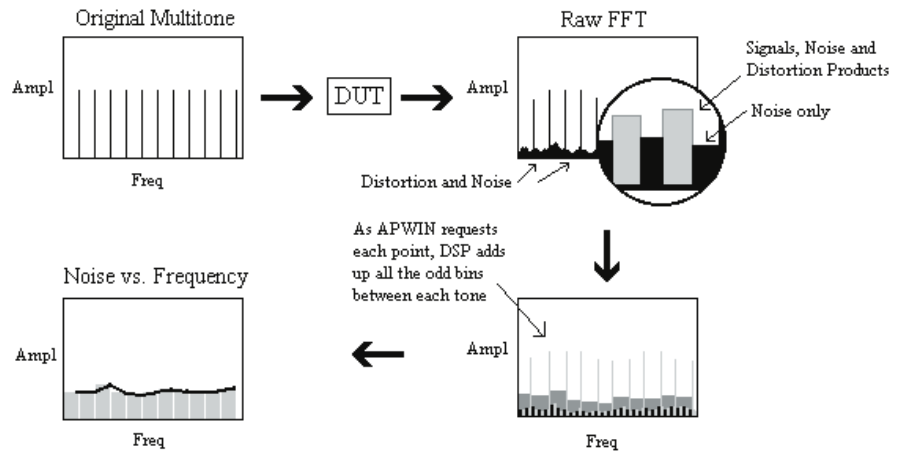


Figure 2-5 Diagram of a Noise Measurement

Don't worry if this is confusing. The point is that if you use a Transform Length that is twice the length of the waveform file, *any frequency you can possibly create will fall into an even-numbered bin in your FFT*. In addition, any distortion products created by your multitone will also fall into even-numbered bins. The only thing that can fall into the odd-numbered bins is the noise created by the device-under-test.

The 'Noise' measurement mode only measures the odd-numbered bins, so it cannot measure signals, only noise. When APWIN asks for a data point, the DSP will add up all the odd-numbered bins between the previous point and the current point (including both endpoints) and send back the total. This will equal the noise between these two frequencies.

If you only ask for two points on your graph, at 20 Hz and 20 kHz, the 20 Hz data point will be meaningless and the 20 kHz point will be the total noise in the audio band. This should give results similar to a conventional noise measurement, except that it will be 3 dB lower because only half the bins are included in the measurement.

A sweep table is not required, since the tones in your multitone do not get in the way of this measurement. Usually setting the sweep panel to the desired endpoints and a certain number of steps in between is sufficient. However, if you use a sweep table you can divide the frequency spectrum any way you want, separating low-frequency noise from high-frequency noise or whatever other divisions you choose.

Hum Measurements

You can also use multitone techniques to test hum. Hum is the result of your power supply line frequency leaking into your audio signal. It is always at either 50 Hz or 60 Hz, depending on your country.

To test hum, you need to use a multitone that does not include a tone at the frequency of your power line or at integer multiples of that frequency.

You will want to check the value of the amplitude bin at the frequency of the power supply line, and at least the 2nd and 3rd multiples of that frequency. Usually the Response measurement mode is used, with a sweep table that contains the values of each bin that is examined for hum products.

Sequence of Events

When learning about multitone testing, it is helpful to know the sequence of events in collecting, preparing and graphing the data. This will help to develop a complete understanding of what we are doing and why we are doing it.

We will start our journey when you press APWIN's 'Go' button, which tells APWIN to collect data and create a graph or table. Before you press Go, you should have all the APWIN panels set up the way you want them and your device-under-test connected and prepared.

When you press Go, APWIN will pass this information to the DSP. The DSP will go into 'Waiting for Trigger' mode and wait for a trigger. A trigger is an event that tells the DSP to begin recording data. There is more about triggering in the chapter on 'Advanced Concepts'.

After the DSP sees its trigger event, it will begin recording each sample that arrives. This activity is known as 'acquisition'. It will continue until a certain number of samples have been recorded. Using APWIN, you can set the number of samples to be recorded during the acquisition. Since the samples arrive at a constant rate, this sets the time interval of the acquisition.

The FFT cannot begin until the acquisition is complete. This is because an FFT needs a large number of samples to work on before it can do its job. The FFT operates on all the samples simultaneously. Once the FFT has begun, you cannot add samples or change the samples, or the results will be incorrect.

Once the acquisition is complete, the FFT will begin. The DSP will compute for a few milliseconds, and then alert APWIN that the FFT is complete. At this point, all the bins from the FFT will be ready and available for graphing.

APWIN will then begin requesting data points for the graph. It will ask for the data at a certain frequency, and the DSP will find the bin containing that frequency and send back the value of that bin. Depending on the measurement mode, the DSP may perform some other computations before sending the data value back to APWIN.

The frequency points that APWIN requests will be determined by the sweep table, or by the settings on the sweep panel (if a sweep table is

not used). There is not necessarily any correspondence between the number of FFT bins and the number of points on the graph. If APWIN asks for several frequencies that are all in the same bin, all these points will have the same value. On the other hand, APWIN may not ask for a point out of every bin, so some bins will not be shown on the graph.

APWIN will plot each point on the graph, drawing a straight line from the previous point. When it has requested all the points specified by the sweep table or sweep panel, the graph will be complete and control will be turned back over to you.

At this point, you may press Go again, which will cause the whole process to repeat. Or, you may press Ctrl+F6, which will repeat the requesting and graphing of data points without going through another acquisition and FFT.

This may seem useless, since the data will be the same as it was last time. True, but only if you leave all the settings the same. Before you press Ctrl+F6, you can change the sweep table and the measurement mode and obtain an entirely different measurement from the same raw data. This is useful for maximum speed - it allows you to get a number of measurements from only one acquisition cycle.

In addition, it minimizes the time that the device-under-test needs to carry the signal. Once the signal has been acquired, any number of tests can be performed on the signal even after the device-under-test has been disconnected.

If you need to, you can even decide whether to run a certain test based on results of previous tests. For example, you can run a Total Distortion test, and if that fails, run a few other tests to determine the type of distortion or problem area, without reacquiring the data.

You can read more about this capability in the chapter on Multiple Measurements.

Introduction to Setup Chapters

The previous chapter described the general application of multitone techniques for audio testing. The next two chapters show you how to use System Two to apply these techniques.

Chapter 4 describes the process for setting up an analog multitone measurement, using analog inputs and outputs. Chapter 5 describes the process for setting up a digital multitone measurement, using digital inputs and outputs.

Unfortunately, we cannot demonstrate analog output to digital input or digital output to analog input without some sort of external device to make the conversion from analog to digital. For those of you that want to set up this type of measurement, it would be a good idea to go through both the digital and analog examples.

Each chapter begins with a flow diagram showing the entire signal path, all the steps in the setup process, and the panels on which all the important settings are located. If you are very familiar with APWIN and your System, these diagrams may be all you need. The flow diagrams show the panels set up for a basic frequency response test.

Following the flow diagram is an overview which describes each step in the setup process according to the flow diagram. This section expands on the flow diagram, offering more detail and descriptions of each important setting.

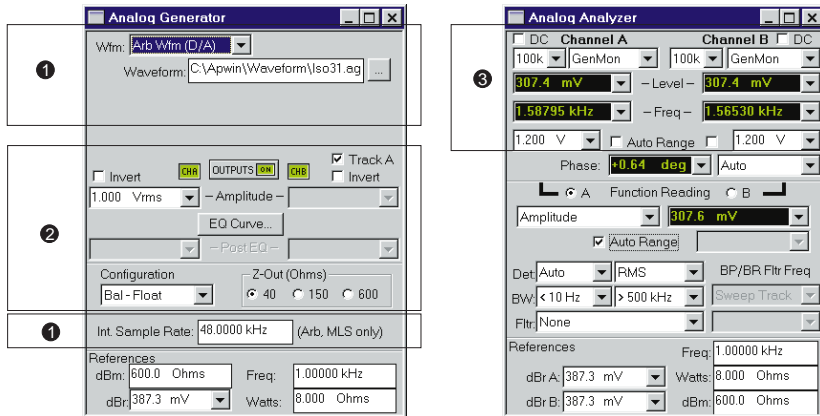
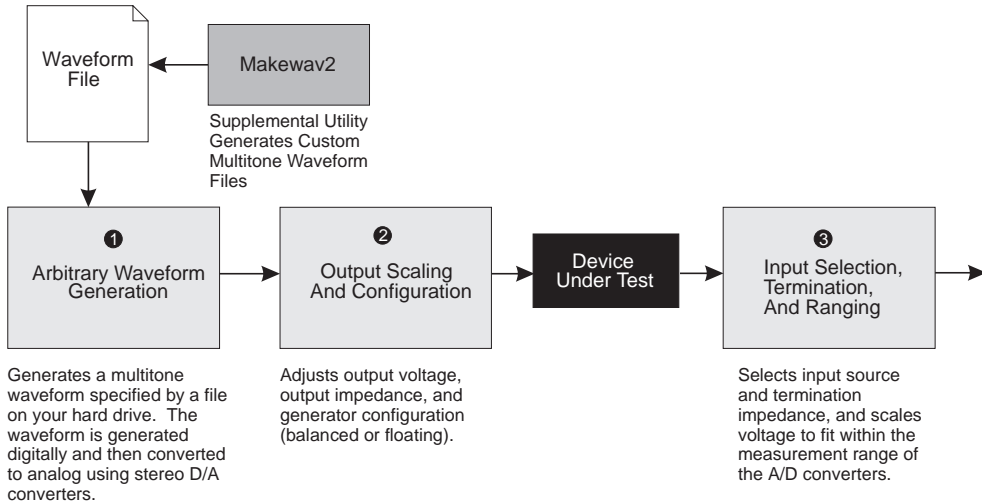
Last in each chapter is a brief tutorial. Each tutorial takes you through the process of setting up a basic response test step-by-step. This gives you a chance to follow the flow diagram and overview and get some hands-on experience setting up each part.

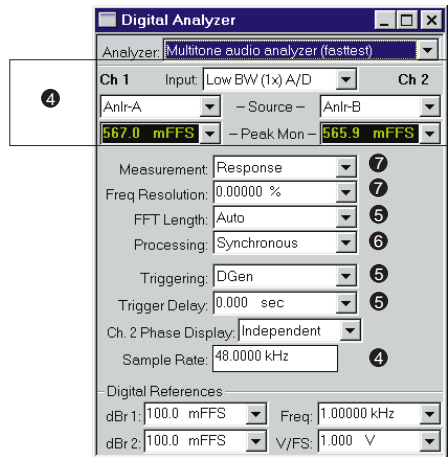
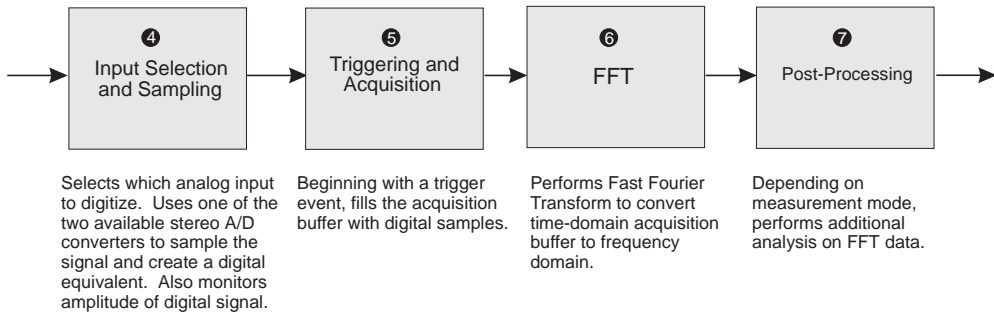
In order for these tutorials to work as described, you will need a PC-compatible computer running APWIN, and a System Two with DSP option. If you would like to do measurements with digital inputs and outputs, you will also need the DIO option. Analog-only systems are not capable of multitone tests, since they have no DSP.

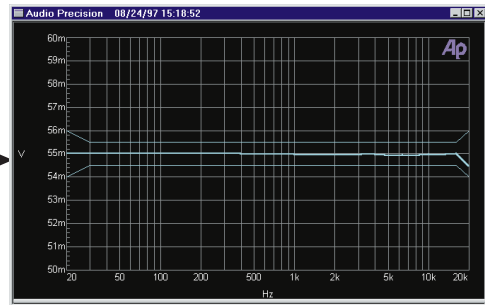
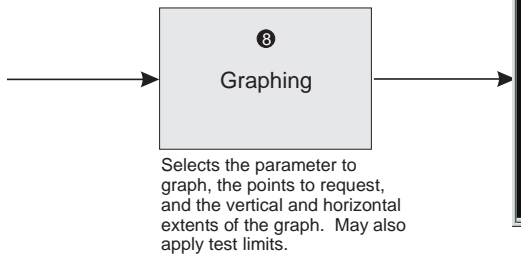
User Notes

Analog Multitone Testing with System Two

Flow Diagram







8

Selects the limit files and data sources (optional)

Selects the points to be requested for the graph

Setup Overview

1

1. Arbitrary Waveform Generation

Generation of the analog multitone waveform is controlled by the Analog Generator panel.

The waveform type (marked 'Wfm:') must be set to 'Arb Wfm (D/A)'. This tells the generator that the waveform will come from a waveform file on your hard drive, which will be generated digitally and then converted to analog using a stereo D/A converter.

The waveform file must specify the multitone you wish to create, and must be in a special file format. These files are created by the Makewav2 program, discussed in Chapter . Some multitone waveform files are also provided with APWIN.

To select a certain waveform file generate, you may type the filename into the box marked 'Waveform', or hit the button just to the right of the box. A window will pop up, allowing you to choose the waveform file from your hard drive.

The sample rate must be set to the rate for which the waveform file was designed. For analog tests, this will almost always be 48k.

2

2. Output Scaling and Configuration

The Analog Generator panel also provides control of the analog characteristics of the output signal. These include on/off control, amplitude, inversion of either channel, bal/unbal, float/gnd, and output impedance. Equalization curves are not supported when arbitrary waveforms are being used.

One or both channels should be turned on, depending on what test is being performed. Generally, neither Invert button will be checked, since the phase of the multitone is specified in the waveform file.

The selection of bal/unbal and, float/gnd and output impedance depend on the type of device you are testing. They should generally be set to match the characteristics of the device-under-test. For most devices, the default setting of Bal-Float and 40 ohm output impedance is satisfactory. The 150 ohm and 600 ohm settings are generally only used when the device-under-test has this same input impedance.

The amplitude of the analog output can be set from the Analog Generator panel, but the actual amplitude will not be the same as the number entered on the panel. This is because the analog generator does not know the actual amplitude of the waveform, since it depends on the samples specified in the file. It will scale the signal as though the file specified a single sine wave of the maximum possible amplitude.

Usually when working with multitones, we are mainly concerned about the peak voltage. This is fairly easy to calculate from the waveform statistics, which are given in the output from Makewav2. If you set the analog output amplitude in Vpk (peak voltage), the actual peak voltage will be less than the requested voltage by the amount of headroom in the multitone waveform. The headroom is given in Makewav2's output, and defaults to 1 dB.

The formula for calculating the peak voltage is this:

$$V_{opk} = V_{pk} \cdot 10^{-\frac{HR}{20}}$$

where V_{opk} is the actual peak analog output, V_{pk} is the peak amplitude selected on the Analog Generator panel, and HR is the headroom given in the output from Makewav2.

Calculating the RMS value of the output waveform is also possible from the waveform statistics, but it is probably easier just to set the analog analyzer inputs to GenMon and measure the amplitude using the input voltmeter. The formula for calculating the actual RMS amplitude is this:

$$V_{out} = V_{rms} \cdot \left(\frac{\sqrt{2}}{CF}\right) \cdot 10^{-\frac{HR}{20}}$$

where V_{out} is the actual analog output voltage, V_{rms} is the amplitude selected on the Analog Generator panel, CF is the crest factor and HR is the headroom. The crest factor and headroom are given in the output from Makewav2.

3

3. Input Selection, Termination, and Ranging

The Analog Analyzer panel provides control of the analog input source, proper termination, and scaling of the analog signal to fit within the measurement range of the A/D converters.

The input source will generally be 'XLR-Bal' or 'BNC-Unbal', to take input from the appropriate front panel connectors. In the setup examples and tutorials given here, the input is 'GenMon', to take the signal directly from the generator, so that external cabling is not necessary.

The termination should be set to match the output characteristics of the device you are testing. For most devices, the default setting of 100k ohms is acceptable. The 600 ohm and 300 ohm settings are generally only used with devices with the same output impedance.

The ranging is required to scale the signal so that it fits within the measurement range of the A/D converters. For best performance, you want the signal to have the highest amplitude possible without reaching the maximum allowable instantaneous amplitude at the A/D's.

The A/D's can measure signals as large as about 3.8 Vrms. The Analog Analyzer's ranging circuitry can attenuate the incoming signal, expanding the measurable range to about 160 Vrms. In addition, the resolution of the A/D converters limits their accuracy at low voltage levels, so the Analog Analyzer can boost the signal as well.

Normally, System Two can sense the amplitude of the signals and range itself for optimal performance. This feature is enabled by the Auto Range check boxes. When either of these boxes is set, its range is automatically selected by System Two. This is called autoranging.

The autoranging can be fooled by certain types of signals and never find a stable range. If this happens, you should lock down the ranges.

Unless the signal you are measuring can have a widely varying maximum amplitude, you should turn autoranging off and select the ranges yourself. This will speed up your tests considerably, since autoranging can take a long time and acquisition cannot begin until the correct range is selected.

To do this, click on the Autorange Check Box. The check mark will disappear, and the Range Entry Field will become active. When it becomes active, it will display a number, which is the highest RMS voltage permitted for the current range. To specify what range you want, type into this box. The input range sets have 13 possible ranges: 160V, 80V, 40V, 20V, 10V, 5V, 2.5V, 1.2V, 600mV, 300mV, 160mV, 80mV, and 40 mV. APWIN will take whatever voltage you type into the box and round it up to the next higher range, and report this range to you.

It is a little bit difficult to tell which range is appropriate for a certain signal, since these ranges are specified in volts RMS and the A/D's care about peak voltages, not RMS. The best way to make sure this is set correctly is to watch the digital signal monitors on the Digital Analyzer panel and make sure that they never touch full scale. You may need to change several settings on the Digital Analyzer panel (as shown in step 4) before these readings will be valid.

If the ranges are stable, and you want to lock down the ranges for optimal speed, you need only to uncheck the Auto Range boxes with the multitone on the input. The range will stay where the autoranging circuitry placed it, which will be correct for the active multitone.

4

4. Input Selection and Sampling

The first thing that must be selected on the Digital Analyzer panel is which analyzer to use. For multitone testing, you should choose 'Multitone audio analyzer (fasttest)'.

In order to obtain an FFT, a number of digital samples must be recorded. These must come from either the digital signal connectors or from the A/D converters. The input to the A/D converters must be the analog signal to be analyzed.

For analog tests, the Input should be set to 'Low BW (1x) A/D', which will take the input from the A/D converters.

The analog inputs to the A/D converters can come from several sources, including a gen-mon path directly from the analog output of the D/A converter and various points within the analog analyzer circuitry. For most tests, the 'Anlr-A' and/or 'Anlr-B' inputs should be used. These selections take the inputs from the analog signal paths,

directly after the ranging circuitry. This is the same point from which the input voltmeter readings are taken. The signal at this point is a scaled version of the input signal, affected only by the ranging and the input impedance selections.

5

5. Triggering and Acquisition

The next step in the process is for the DSP to acquire a number of digital samples and store them in a special area of memory called the *acquisition buffer*. You can control the length of the acquisition buffer (how many samples will be acquired) by changing the FFT Length.

The FFT Length option is discussed in detail in the chapter on Advanced Options. In general, the tradeoff is one of speed vs. resolution. A longer acquisition buffer will take longer to acquire and to perform the FFT, but it offers higher frequency resolution. Unless speed is critical, 'Auto' is usually used, which will select an acquisition length that is twice the length of the generator waveform.

The acquisition will not begin until a certain trigger condition exists. The triggering options are described in detail in the chapter on Advanced Options. The 'Free Run' trigger condition is always true; it causes an immediate acquisition. It may be used whenever the timing of the acquisition is not critical, and the signal is present at the time you press 'Go'.

If synchronous processing is selected, the acquisition will begin simultaneous with the first sample of the generator waveform. This will assure that an integer number of generator waveform cycles are captured, as long as the acquisition is at least as long as the generator waveform.

See the chapter on Advanced Options for a complete discussion of windows, frequency correction, and synchronous processing.

6

6. Fast Fourier Transform

After acquiring a buffer full of digital sample data, the DSP will perform the Fast Fourier Transform, or FFT. This mathematical process converts the time-domain samples into a set of frequency-domain bins. Each bin contains the combined amplitude of all sine wave components within a range of frequencies. The number of bins is

always equivalent to half the number of samples originally obtained, plus one. The frequency range from 0 Hz to half the sample rate is divided linearly into the set of bins.

As part of the FFT process, a *window* may be applied to the data. A window is almost never used in multitone testing because the effects of a window render most multitone analysis techniques invalid. Synchronous or frequency-corrected processing is preferred. The Processing selection makes the choice between windowed, synchronous, and frequency corrected processing.

In analog testing, synchronous processing can only be used when the same System Two is simultaneously generating and analyzing the multitone. If a different System or the device-under-test is generating the multitone, frequency correction must be used to align the signal so that the acquisition buffer contains an integral number of generator waveform cycles. Freq Corrected Processing must be selected, and the Freq Resolution must be set to the maximum allowable frequency error. Frequency correction can only adjust for about 3% frequency error, although the Freq Resolution can be set to higher values for use during post-processing.

See the chapter on Advanced Options for a complete discussion of windows, frequency correction, and synchronous processing.

7

7. Post-Processing

After performing the FFT, some other analysis is performed on the FFT bins, depending on the measurement desired. This analysis depends on the selected Measurement. The various measurement modes are described in the Concepts chapter, along with how they are used to create each type of measurement.

The post-processing generally occurs on a point-by-point basis, as APWIN requests data points from the DSP.

The Freq Resolution setting is used when a device modulates (or varies the frequency of) the multitone signal. For example, an analog tape recorder may create slight speed changes, which will result in modulation of the multitone's frequencies. This will cause the tones in the multitone to spill over into adjacent bins, creating sidebands, which can be very critical to multitone tests.

If your device causes frequency modulation, the Freq Resolution should be set to the maximum amount of frequency modulation expected from the device-under-test. This allows the post-processing calculations to accommodate the modulation.

Note that this is different from frequency correction. Frequency correction adjusts for a static frequency shift, where the frequency of the acquired multitone has a stable frequency which is different from the System Two's generation frequency. Frequency resolution adjusts for variations of the multitone frequency within the acquisition time. The Freq Resolution setting, however, affects both operations.

A complete discussion of the Freq Resolution setting is given in the chapter on Advanced Options.



8. Graphing

The final stage in the process is to set up the graphing options so that the correct graph is obtained. This setup takes place on the Sweep panel.

The first thing you will want to select is the Source1. This should be set to Fasttest.FFT Freq. The Start frequency should be set to the desired frequency at the left-hand edge of the graph. The Stop frequency should be set to the desired frequency at the right-hand edge of the graph.

You will also need to select the Data1. For most measurements, you should choose either Fasttest.Ch.1 Ampl or Fasttest.Ch.2 Ampl, depending on which channel you want to use. If you want both channels to be graphed simultaneously, you can select one for Data1 and the other for Data2.

The only time you won't use the Ampl parameters is when doing phase testing. For these types of tests, you will want to use the Fasttest.Ch 2 Phase selection, since this will give the interchannel phase reading. Make certain that the Ch 2 Phase Display setting on the Digital Analyzer panel is set to 'Interchannel'.

The most critical selection is the choice of which points to include in the graph. This is often very important multitone tests, since choosing the wrong points may result in very different data than expected. Usually it will be desirable to request the points that are present in the

original multitone. Different measurements and measuring situations may require different sets of points to graph.

The points to graph must first be stored in a sweep table file, which usually has the extension ADS. The sweep table files are usually created from data generated by Makewav2 at the same time as the multitone waveform file is generated. See Chapter for instructions on how to do this.

Sweep tables may also be created directly by editing the data values in the data editor and then saving the data file as a sweep table, using the menu item File Save As Sweep Tables.

Like all APWIN data files, the sweep table files are arranged in columns of data. In sweep table files created using Makewav2, the data desired for graphing is contained in the first column. If you create your own sweep table files, you may put the data in any column, but remember which column you used so that you may select it later. The first column is usually the best choice.

You also must set the units to a frequency-compatible unit, such as Hz. The units may not be changed in the data editor. You must go back to the Sweep panel and change the parameters so that the required units are allowed. The Source1 fields correspond to the first data column, the Data1 fields correspond to the second data column, and the Data2 fields correspond to the third data column.

It is best to select the correct units before using the data editor.

In many cases the data editor will be full with data from a previous sweep or may contain no data values at all. Pressing the right mouse button will bring up a menu containing the options to add or delete rows of data. To delete many rows, select the rows by dragging across the row-number buttons on the left edge of the data editor while holding down the left mouse button. Then a delete operation will affect all the highlighted rows.

When you're done creating your sweep table, be sure to save it as a sweep table using File Save As Sweep Tables.

After creating the sweep table either by hand or using Makewav2, you must select it as the sweep table for the graph to use when selecting which points to request for the graph. To do this, press the Table

Sweep button on the Sweep panel. Another window will pop up, allowing you to choose the sweep table file either by typing the name of the file into the File box or pressing the browser button to the right of the File box. The File box also drops down to provide a list of recently-used sweep tables and the option of 'None'. Choosing 'None' will disable the table sweep function, and the points to be requested will be calculated linearly or logarithmically using the normal fields on the Sweep panel.

You must also select the column (in the sweep table file) from which to take the point data. This will usually be column 1, unless you used a different column when creating the sweep table by hand.

The Sweep panel also allows selection of upper and lower acceptance limits for each of the two data sources. These choices operate very similarly to the sweep table selections, except that the files have the extension ADL and must be saved using **File Save As Limit Data**. Limit files cannot be generated by Makewav2, so they must be edited (usually from an initial run of the test) using the data editor.

Tutorial - Generating a Multitone

In order to make any multitone measurement, we first need to set up the signal generator to output a multitone.

Let's start with a fresh test, so that we have a common starting point. To do this, choose **File New Test** from the menu or press the New Test Button, shown on the left.



New Test Button



Analog Generator Button

The generator settings are made on the Analog Generator panel. This panel is usually visible immediately after pressing New Test. If you don't see it, select **Panels Analog Generator** from the menu or press the Analog Generator Button, shown on the left.

The generator panel should look like this:

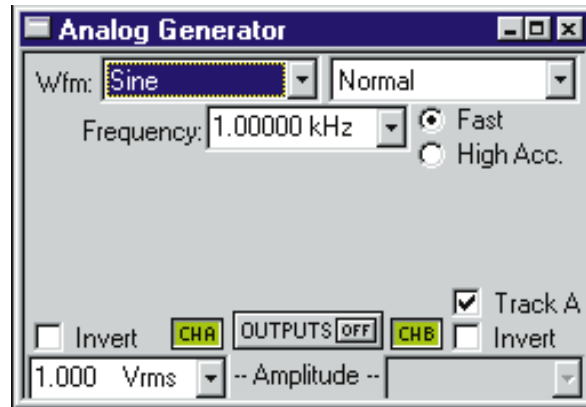


Figure 4-1 Analog Generator Panel, Small Version

This panel controls all the attributes of the signal generated by the generator.

The first thing we need to do is select a multitone waveform using the 'Wfm:' choose box. If you press on the down arrow, you will see a list of all the available waveforms that the System Two can generate. Some of these are generated using dedicated analog hardware, while others are created digitally using a DSP and then converted into an analog signal using a D/A, or Digital-To-Analog converter. The digitally-generated ones are marked '(D/A)'.

In order to generate a multitone waveform, the choice we need is 'Arb Wfm (D/A)'. This stands for 'Arbitrary Waveform', generated by the DSP. Make this selection now.

Next, we need to tell the System Two where to find the waveform file. For this example, we will use a waveform file provided with APWIN. It is called 'Iso31.agm'. If you installed the software in the default locations on your hard drive (you didn't change the installation directories), this should be in C:\Apwin\Waveform. If you did change the installation directories, the exact path may be different.

To load this file into the arbitrary waveform generator, press the 'Choose Waveform' button, shown on Figure 4-2.

Note: If there is no Choose Waveform button, then 'Arb Wfm (D/A) is probably not selected in the Waveform List box (marked 'Wfm').

After you hit the Choose Waveform button, you will see a normal file-choosing window. Find your way to the directory where Iso31.agm is found and choose it. Once you have chosen the file, the file-choosing window will go away and the Analog Generator panel will show the name of the file in the box marked 'Waveform'. If the filename and path are too long, only the first part will be displayed.

The generator is now set to generate the multitone specified in Iso31.agm. This waveform contains 31 different tones, all produced simultaneously.

Tutorial - Generator Settings

The generator is probably turned off at the moment. To turn it on, press the Output On/Off button, marked on Figure 4-2. You can also independently control channel A and channel B with the green buttons that are on either side of the Output On/Off button. Set all the buttons so that they are green. This will turn both channels on.

The Analog Generator panel should now look like this:

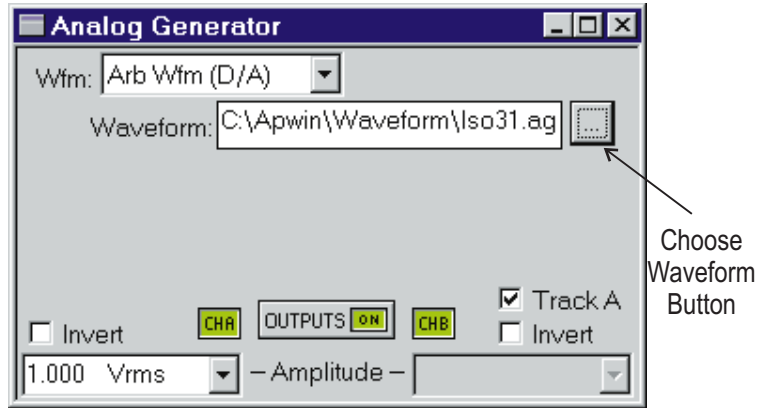


Figure 4-2 Analog Generator Panel with Multitone



Let's listen to the generator waveform. Choose Panels Headphone/Speaker from the menu, or hit the Speaker Button, shown on the left.

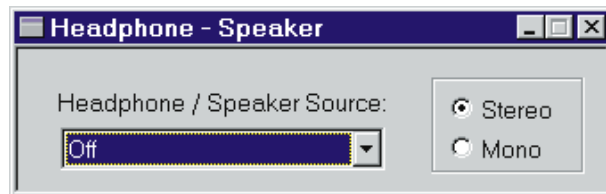


Figure 4-3 Headphone/Speaker Panel

You will see a new panel which allows you to hear the signals at various points within your System Two. It should look like this:

Turn down the speaker knob (fully counterclockwise) on the front of your System Two, and change the choose box on the Speaker panel to 'Generator Monitor'. This tells the speaker to reproduce the sound created by the generator. Then turn up the speaker knob to a comfortable listening level. You should hear a very rich, organ-like tone. This is the sound specified by the file Iso31.agm.

Tutorial - Analog Analyzer Settings

By now, our generator should be making a nice Iso31 multitone.

The signal to be analyzed must first pass through the Analog Analyzer. This is where it will be properly terminated and scaled. After the signal has been scaled to fit within the range of the A/D's, it will be routed to the A/D's to be sampled.



Analog Analyzer
Button

If the Analog Analyzer panel is not visible, choose Panels Analog Analyzer from the menu or hit the Analog Analyzer Button, shown on the left.

You should see a panel that looks like this:

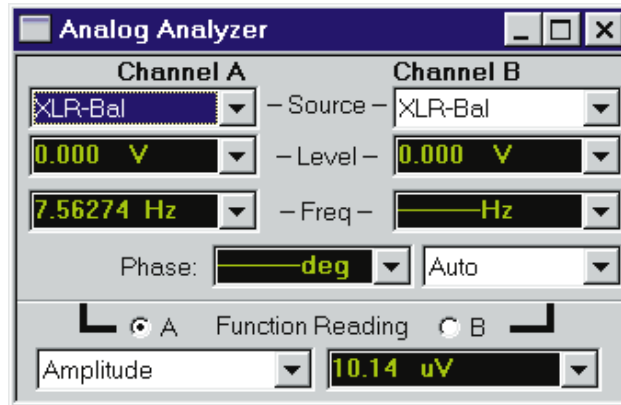


Figure 4-4 Analog Analyzer Panel, Small Version

The numbers will be different than what is shown here, and will probably be changing as you watch.

The first thing you will want to change is the Source Selection. This determines which front panel connectors you are using for input. Set the channel A and channel B Source Selections to 'GenMon'. This stands for 'Generator Monitor'. The System Two has an internal signal path directly from the generator to the analyzer, and setting the Source Selection to GenMon will take the signal from this path instead of from the connectors on the front panel of the instrument.

This panel also allows us to monitor the input signals to make sure they are present and approximately correct. The 'Level' and 'Frequency' readings for each channel come directly from the selected inputs.

Once you have selected Gen-Mon for the input source, the Level meters should show a voltage near 300mV. The frequency meters read the frequency of the incoming signal, but they won't work correctly when observing a multitone waveform. These frequency meters are designed to operate on a signal with one clearly dominant frequency. Since the Iso31 multitone contains numerous frequencies at about the same level, the frequency reading from these meters will be erratic and meaningless. Similarly, the phasemeter will not correctly analyze a multitone.

Why only 300 mV?

You may be wondering why we read only 300 mV when the generator is set to output 1 V. This is because the Analog Generator is calibrated to the peak value of the output signal, and the Analog Analyzer is reading the value in RMS. The Analog Generator will scale the output so that the peak value of the multitone will be the same as the peak value of a sinewave with the selected RMS amplitude. Since the multitone has a high crest factor (the ratio of the peak value to the RMS value), its RMS value will be significantly less than a sinewave with the same peak value.

See the previous section on Output Scaling and Configuration for more detail about calculating the actual output voltage.



Digital Analyzer
Button

Tutorial - Digital Analyzer Settings

Now that the System Two is set up to generate and prescale the signal, we need to make the settings for sampling and processing the FFT. This is done on the Digital Analyzer panel. To see the Digital Analyzer panel, choose Panels Digital Analyzer or hit the Digital Analyzer Button, shown on the left.

The Digital Analyzer panel controls the A/D's and also the DSP, which does the analysis of the sampled data. There are several Analyzers for the DSP, which enable different types of measurements. By default, the Digital Analyzer is disabled. You can select an Analyzer by

choosing one of the measurement functions in the drop-down box in the middle of the panel.

Choose the program that says 'Multitone audio analyzer (fasttest)', and the panel should change so that it looks like this:

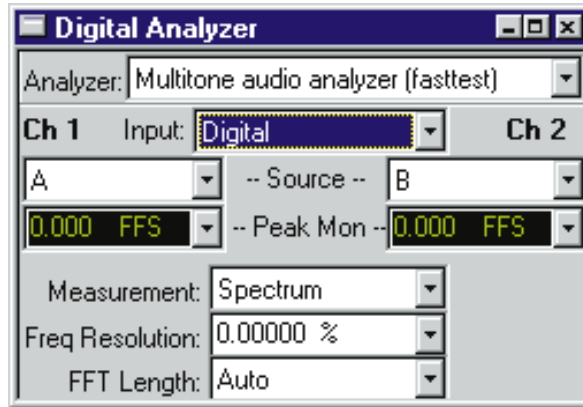


Figure 4-5 Digital Analyzer Panel, FastTest Version

The first thing we need to do is change the Input Type, marked 'Input:'. The DSP can be used to measure digital signals plugged into the digital inputs, as well as analog signals sampled with an A/D. Press the down arrow on this field and select 'Low BW (1x) A/D'.

Once you have selected an A/D measurement as the Input Type, the Source Selector choose boxes will let you choose among the available analog signal sources. Each of these choices takes a signal from somewhere else inside your System Two. To do multitone measurements on the analog input signals, choose Anlr-A for the channel 1 source and Anlr-B for the channel 2 source. These signals come from the analog analyzer, after prescaling. This will be the same signals being measured by the level meters at the top of the Analog Analyzer panel.

Press the down arrow to the right of the box marked 'Measurement' and choose the Response measurement mode.

One important feature of this panel is the Peak Monitors, which display the peak values of the signals at the A/D converters. As we mentioned earlier, in order to get good data we must be certain that the input to

the A/D's never exceeds the voltage that they can process. These meters allow us to monitor the peak values obtained by the A/D's.

The default units are 'FFS', which means 'Fraction of Full Scale'. Full Scale is the maximum voltage that the A/D's can correctly analyze. This is represented as 1.000 FFS. So, when you are setting up your FFT you need to be certain that the signal is scaled such that these peak meters always read values below 1.000 FFS.

Tutorial - Graph Setup

At this point, we are generating a good multitone, sending it to the analyzer, prescaling it, and sending it to the A/D's to be sampled. The Digital Analyzer is prepared to perform an FFT and give us our desired points. All we have left to do is set up the graphing.

To set up graphs, we need the sweep panel, so select **Panels** **Sweep** from the menu or hit the Sweep Button, shown on the left.



Sweep Button

You should see a panel that looks like this:

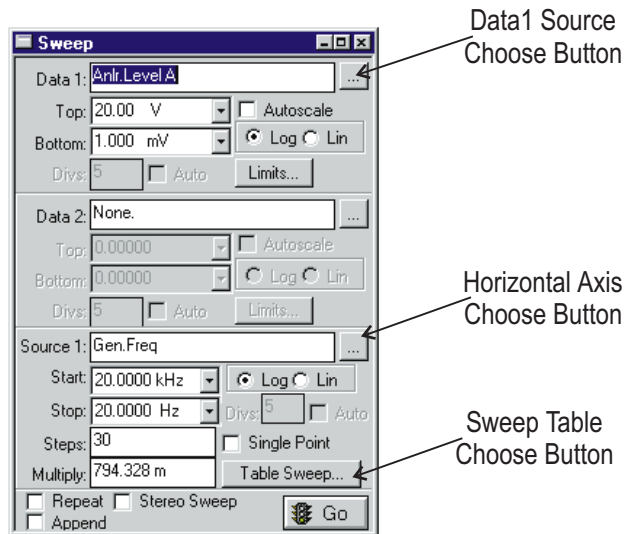


Figure 4-6 Sweep Panel, Small Version

This is the panel that APWIN uses to run a sweep or create a graph.

The first thing we need to change is the horizontal axis parameter. Hit the Horizontal Axis Choose Button (marked on Figure 4-6), and a window will pop up that looks like this:

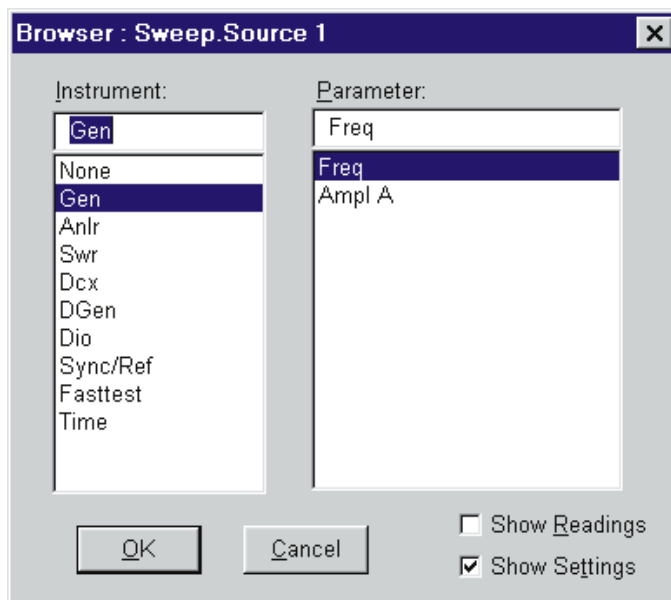


Figure 4-7 Horizontal Axis Browser

This window allows you to choose a setting from any of the ‘instruments’ within your System Two. Choose the Fasttest instrument, and then choose the ‘FFT Freq’ setting. Then hit OK.

Note: The window for choosing the sweep source depends on what settings are available. If you don’t see the settings you want, check the Digital Analyzer panel. Make sure the analyzer is set to ‘Multitone audio analyzer (fasttest)’.

The next thing we need to change is the Data 1 Source. Hit the Data 1 Source Choose Button (marked on Figure 4-6), and a window will pop up that will allow you to choose the data to use for the first trace. Choose the Fasttest instrument, and then choose ‘Ch.1 Ampl’. Press ‘Ok’ and then set the Top of the Graph to 1 V and the Bottom of the Graph to 100 nV.

Since we want frequency along the bottom of the graph, and we want it to go from 20 Hz to 20 kHz, we don't need to change the 'Start' and 'Stop' Fields.

It is not entirely necessary to use a sweep table in this application, but it will provide the most accurate representation of the data. To do this, we use the Table Sweep feature to take data from only those frequencies that existed in the output waveform.

Press the Sweep Table Choose Button on the Sweep Panel (marked on Figure 4-6). You should see a window that looks like this:

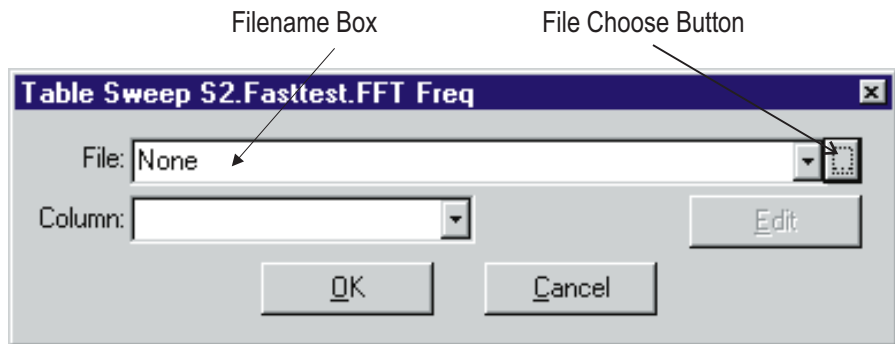


Figure 4-8 Sweep Table Setup Window

This window is where we define the points that will be used for the graph. The points must be taken from a data file somewhere on your hard disk. For this example, we will use ISO31.ads, which is provided with your APWIN software. ISO31.ads is specifically designed to go with ISO31.wav.

Press the File Choose button (marked on Figure 4-8). You will see a standard window allowing you to choose a data file. Choose ISO31.ads, which is located under \Waveform in the directory where you chose to install the data files. If you used the default locations when installing APWIN, then the full pathname will be C:\Apwin\Waveform\ISO31.ads

If you like, you may type the full name into the Filename Box (marked on Figure 4-8) instead of pushing the File Choose Button.

Once you have selected the file, the Column Box will read "1=S1.Fasttest.FFT Freq". This does not need to be changed.

The Sweep Table Setup Window should now look like this:



Figure 4-9 Sweep Table Setup Window

Now hit Ok to close the Sweep Table Setup Window. When you return to the Sweep panel, you should see that the Steps and Multiply boxes are gone. This is because the Table Sweep actually replaces these fields.

Now press the Go/Stop button on the Sweep panel, and you should see a graph that looks like this:

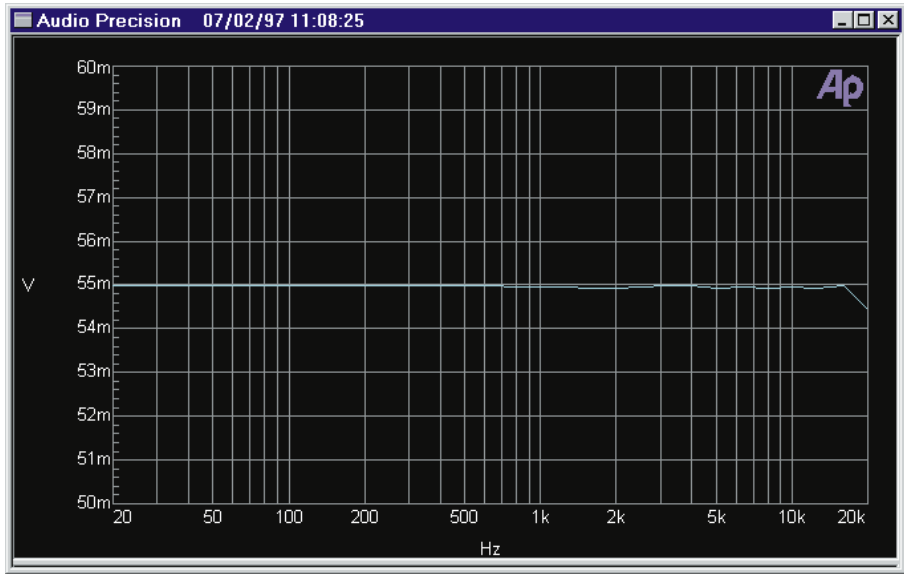


Figure 4-10 Response Graph

This is the response graph we have been working to create, but it is on too large a scale to be very useful.

Go back to the Sweep panel. Change Data1 Top to 60 mV and the Data1 Bottom to 50 mV. The graph should change to the following:

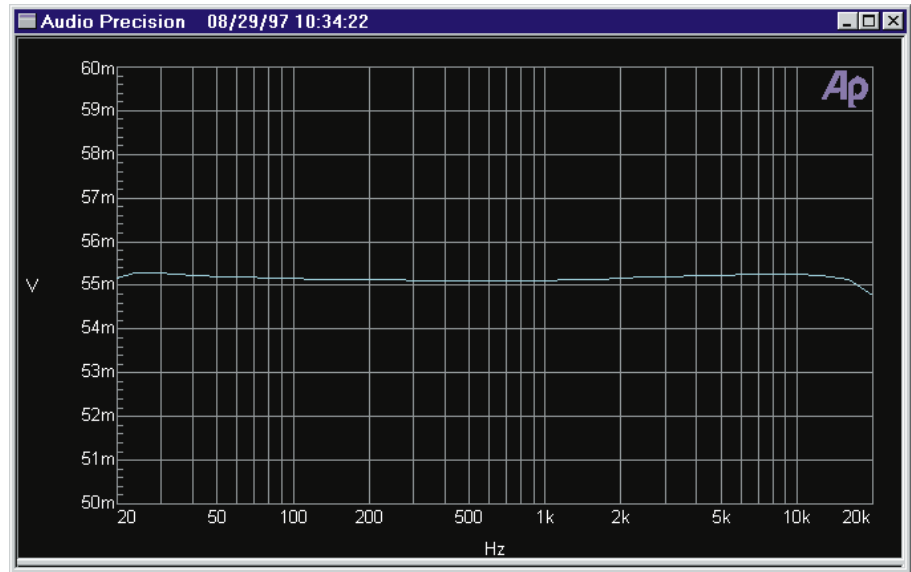


Figure 4-11 Response Graph, Zoomed In

This is a more useful view of the data.

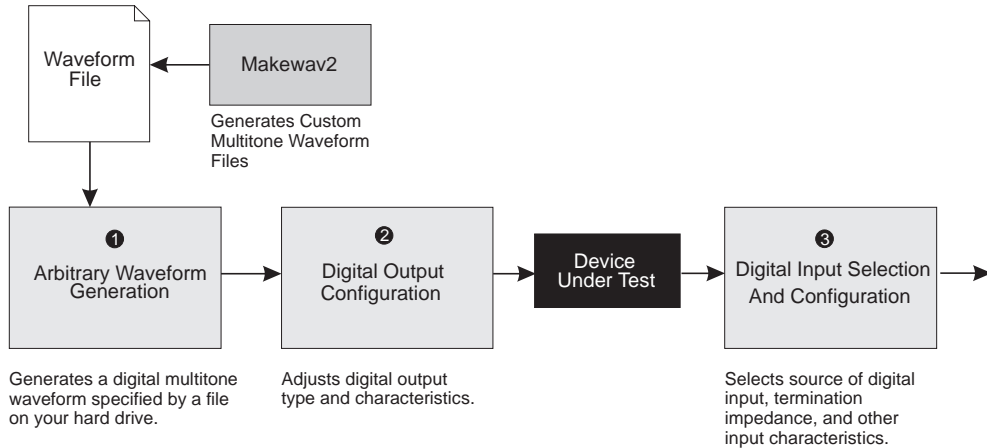
If you were testing a real device, you would probably want to apply limits to the data, or change to a dB-based unit which would allow easier interpretation of the flatness data. Both of these can also be accomplished using the Sweep panel.

This concludes this exercise in setting up an analog multitone response graph for System Two.

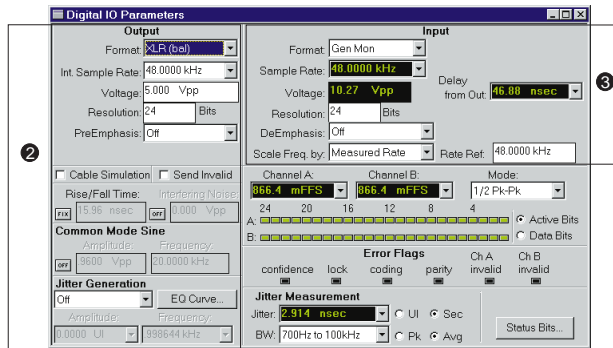
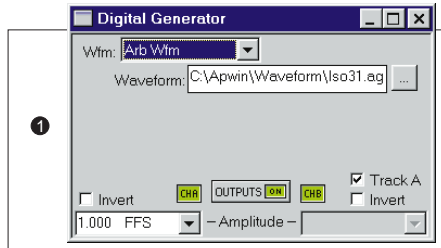
User Notes

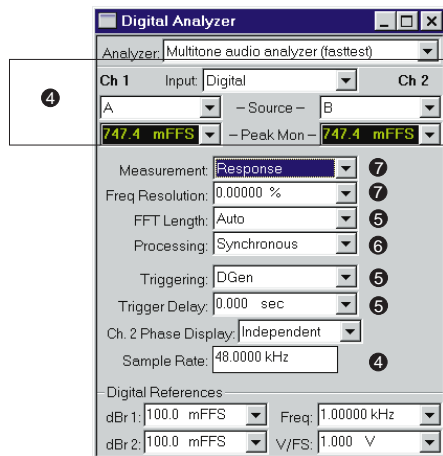
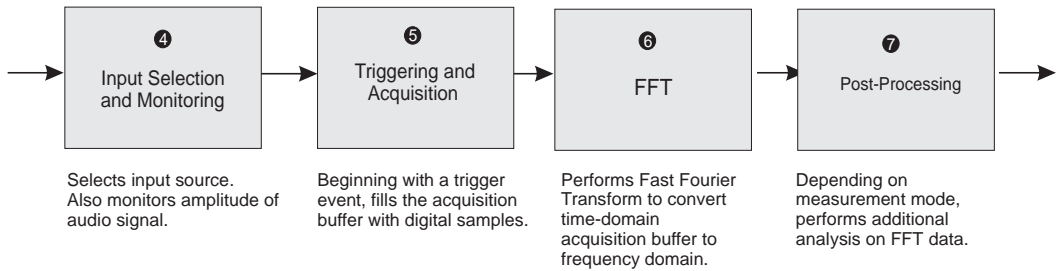
Digital Multitone Testing with System Two

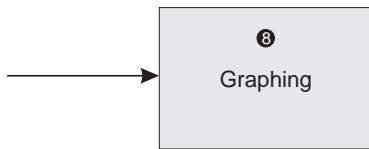
Flow Diagram



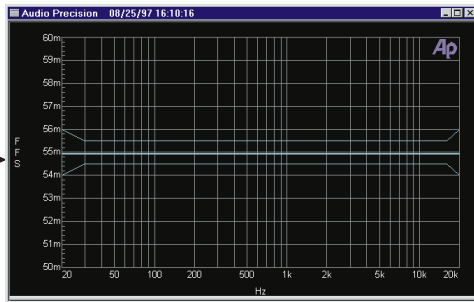
5 S2 Digital







Selects the parameter to graph, the points to request, and the vertical and horizontal extents of the graph. May also apply test limits.



8

Selects the limit files and data sources (optional)

Selects the points to be requested for the graph

Setup Overview

1

1. Arbitrary Waveform Generation

The multitone waveform must originate from the digital generator, which is controlled by the Digital Generator panel.

The multitone waveform is specified by an AGM or AGS file on your hard drive. These files are created by the Makewav2 program, discussed in Chapter . Some multitone waveform files are also provided with your APWIN software.

Before selecting the waveform, the digital generator must be told to generate an arbitrary waveform instead of any of the other waveforms that it can generate. This is done by setting the Wfm to 'Arb Wfm'.

Once you have selected to generate an arbitrary waveform, you must select the waveform file from which to generate. You can type the name of the waveform file into the text box marked 'Waveform', or hit the button just to the right of it. A window will pop up allowing you to choose a waveform file.

There are also controls on the Digital Generator panel for on/off and amplitude of the digital multitone. The on/off controls will generally be set so that they are all green, so that all channels are enabled. The amplitude will generally be set to 1 FFS.

2

2. Digital Output Configuration

The Digital IO panel allows configuration of the digital output.

The Output Format can be set to XLR, BNC, Optical, Serial or Parallel. This should be set to whichever interface you are using to connect to your device-under-test. For Gen-Mon tests, any of the options can be used except the Parallel output format.

The Internal Sample Rate should be set to the sample rate for which the multitone waveform file was designed. Waveform files can be used at other sample rates, but all the component frequencies will be scaled at the same ratio as the sample rate change.

The Voltage that is selected on the Digital IO panel is not the amplitude of the audio signal, but rather the voltage of the digital waveform. It should be set to an appropriate voltage for the device-under-test.

The Resolution setting should be set to match the bits of resolution supported by the device-under-test. Most devices support 16-bit resolution. If more bits of resolution are provided than are supported by the device-under-test, usually the device will ignore the extra bits. However, the dither will be at the wrong level and will not perform its function properly.

The rest of the Output settings on the left-hand side of the Digital IO panel are for special test conditions and signal impairment. See the APWIN for System Two User's manual for a more complete description of each setting.

3

3. Digital Input Selection

The Digital IO panel also provides selection of the digital input format and configuration.

The Format should be set to whichever connector you are using to connect to your device-under-test. The settings which say 'w/EQ' have an equalization circuit that compensates for the 'Cable Simulation' generator setting, so the two are generally only used in conjunction.

If you are using the BNC or XLR connectors, the Z-in should be set to the impedance expected by the device-under-test.

The Resolution should be set to the number of bits of resolution supported by the device-under-test. Most devices support 16 bits of resolution.

The DeEmphasis should generally be set to 'off' unless the device-under-test is using PreEmphasis.

The Scale Freq By should usually be set to 'Measured Rate'. The other options are discussed in the APWIN for System Two User's Manual.

4

4. Input Selection and Monitoring

The first thing that must be selected on the Digital Analyzer panel is which analyzer to use. For multitone testing, you should choose 'Multitone generator/analyzer (fasttest).

The Input must be set to 'Digital' to take the input data from the digital signal path instead of from the A/D convertors.

The digital signal path supports two separate channels, as does the FFT. Either of the two input channels may be sent to either FFT channel using the Source choices.

Once you have chosen the correct inputs, you can monitor the amplitude of the digital audio signal using the Peak Monitors. An appropriate signal level at these monitors should indicate that your other settings and signal routing options are correct.

5

5. Triggering and Acquisition

The next step in the process is for the DSP to acquire a number of digital samples and store them in a special area of memory called the *acquisition buffer*. You can control the length of the acquisition buffer (how many samples will be acquired) by changing the FFT Length.

The FFT Length option is discussed in detail in the chapter on Advanced Options. In general, the tradeoff is one of speed vs. resolution. A longer acquisition buffer will take longer to acquire and to also to perform the FFT, but it offers higher frequency resolution. Unless speed is critical, 'Auto' is usually used, which will select an acquisition buffer that is twice the length of the generator waveform.

The acquisition will not begin until a certain trigger condition exists. The triggering options are described in detail in the chapter on Advanced Options. The 'DGen' trigger condition occurs on each cycle of the generator waveform. It may be used whenever the timing of the acquisition is not critical, and the same System Two is doing the generation and analysis.

6**6. Fast Fourier Transform**

After acquiring a buffer full of digital sample data, the DSP will perform the Fast Fourier Transform, or FFT. This mathematical process converts the time-domain samples into a set of frequency-domain bins. Each bin contains the combined amplitude of all sine wave components within a range of frequencies. The number of bins is always equivalent to half the number of samples originally obtained, plus one. The frequency range from 0 Hz to half the sample rate is divided linearly into the set of bins.

As part of the FFT process, a *window* may be applied to the data. A window is generally only used when the acquisition cannot be handled synchronously with the multitone generation or by frequency correction.

If the generation and acquisition are both occurring simultaneously on the same System Two, synchronous processing is usually used. If the signal originates in a different System Two or the device-under-test, frequency correction can generally be used. See the chapter on Advanced Options for a complete discussion of windows, synchronous processing, and frequency correction.

7**7. Post-Processing**

After performing the FFT, some other analysis is performed on the FFT bins, depending on the measurement desired. This analysis depends on the selected Measurement. The various measurement modes are described in the Concepts chapter, along with how they are used to create each type of measurement.

The post-processing generally occurs on a point-by-point basis, as APWIN requests data points from the DSP.

The Freq Resolution setting is generally used when a device modulates (changes the frequency of) the multitone signal. For example, an analog tape recorder may create slight speed changes, which will result in modulation of the multitone's frequencies. This will cause the tones in the multitone to spill over into adjacent bins, which can be very critical to multitone tests.

The Freq Resolution should be set to the maximum amount of frequency deviation (or modulation) expected from the

device-under-test. This allows the post-processing calculations to accommodate the frequency modulation. A complete discussion of the Freq Resolution setting is given in the chapter on Advanced Options.

8

8. Graphing

The final stage in the process is to set up the graphing options so that the correct graph is obtained. This setup takes place on the Sweep panel.

The first thing you will want to select is the Source1. This is usually set to Fasttest.FFT Freq. The Start frequency should be set to the desired frequency at the left-hand edge of the graph. The Stop frequency should be set to the desired frequency at the right-hand edge of the graph.

You will also need to select the Data1. For most measurements, you should choose either Fasttest.Ch.1 Ampl or Fasttest.Ch.2 Ampl, depending on which channel you want to use. If you want both channels to be graphed simultaneously, you can select one for Data1 and the other for Data2.

The only time you won't use the Ampl parameters is when doing phase testing. For phase tests, you will want to use the Fasttest.Ch 2 Phase selection, since this will give the interchannel phase reading. Make certain that the Ch 2 Phase Display setting on the Digital Analyzer panel is set to 'Interchannel'.

A very critical selection is the choice of which points to include in the graph. Choosing the wrong points may result in very different data than expected. Usually you will use a sweep table to tell APWIN which points to choose.

The points to choose depend on the measurement desired and the composition of the multitone. In most cases, you will use a sweep table that specifies the same points as the tones in your multitone.

The sweep table must first be stored in a sweep table file, which usually has the extension ADS. Then you must press the Table Sweep button on the Sweep panel to choose the sweep table file and the column of data containing the points to choose.

The sweep table files are usually created from data generated by Makewav2 at the same time as the multitone waveform file is generated. They may also be created directly by editing the data values in the data editor and then saving the data file as a sweep table, using the menu item File Save As Sweep Tables.

Like all APWIN data files, the sweep table files are arranged in columns of data. In sweep table files created by Makewav2, the data desired for graphing is contained in the first column. If you create your own sweep table files, you may put the data in any column, but remember which column you used so that you may select it later. The first column is usually the best choice.

You also must set the units to a frequency-compatible unit, such as Hz. The units may not be changed in the data editor. You must go back to the Sweep panel and change the parameters so that the required units are allowed. The Source1 fields correspond to the first data column, the Data1 fields correspond to the second data column, and the Data2 fields correspond to the third data column. It is best to select the correct units before using the data editor.

In many cases the data editor will be full with data from a previous sweep or may contain no data values at all. Pressing the right mouse button will bring up a menu containing the options to add or delete rows of data. To delete many rows, select the rows by pressing the row-number button on the left edge of the data editor. Then any delete operations will affect all the highlighted rows.

When you're done creating your sweep table, be sure to save it as a sweep table using File Save As Sweep Tables.

After creating the sweep table either by hand or using Makewav2, you must select it as the sweep table for the graph to use when selecting which points to request for the graph. To do this, press the Table Sweep button on the Sweep panel. Another window will pop up, allowing you to choose the sweep table file either by typing the name of the file into the File box or pressing the browser button to the right of the File box. The File box also drops down to provide a list of recently-used sweep tables and the option of 'None'. Choosing 'None' will disable the table sweep function, and the points to be requested

will be calculated linearly or logarithmically using the normal fields on the Sweep panel.

You must also select the column in the sweep table file from which to take the point data. This will usually be column 1, unless you used a different column when creating the sweep table by hand.

The Sweep panel also allows selection of upper and lower acceptance limits for each of the two data sources. These choices operate very similarly to the sweep table selections, except that the files have the extension ADL and must be saved using File Save As Limit Data. Limit files cannot be generated by Makewav2, so they must be edited (usually from an initial run of the test) using the data editor.

Tutorial - Generating a Multitone

We first need to set up the digital generator to output a multitone. We will use a multitone that is provided with your APWIN software. If you did not install the samples when you installed APWIN, you will probably want to rerun APWIN Setup and install them now.



New Test Button



Digital Generator Button

Let's start with a fresh test, so that we have a common starting point. To do this, choose **File** **N**ew Test from the menu or press the New Test Button, shown on the left.

The digital generator is controlled by the Digital Generator panel. To make this panel visible, select **P**anels **D**igital **A**nalyzer from the menu or press the Digital Generator Button, shown on the left.

You should see a panel that looks like this:

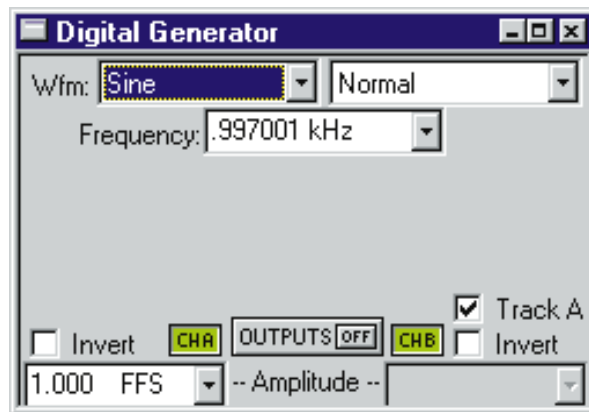


Figure 5-1 Digital Generator Panel, Small Version

The first thing we need to do is set up the generator to output a multitone. The 'Wfm:' selector is used to choose what type of waveform is being generated.

Press the down arrow to the right of the Wfm: box, and you will see a list of the available waveforms. Choose 'Arb Wfm', which stands for 'Arbitrary Waveform'. This will allow us to generate a multitone specified by a waveform file on your hard drive.

The waveform file we will use is called Iso31.agm, and is located in the APWIN data directory, under \Waveform. If you installed using the

default pathnames, the entire path should be C:\APWin\Waveform\Iso31.agm. If you chose not to install the samples, this file will not be installed, so you will want to rerun APWIN Setup and tell it to install the samples.

Press the Choose Waveform button, and you will see a normal file-choosing window. Find your way to Iso31.agm and choose it. Double-click on it, or click on it once and press 'Open' and it will be loaded into the digital generator. When you get back to the regular panel, you should see that the Wfm: box on the Digital Analyzer panel has changed so that it now shows the name of this file, with the entire path. If the path is too long to fit in the box, you may see only the first part of it.

The Output On/Off buttons are used to turn the output on and off. The 'Outputs' button controls both channels, and the other two buttons control each channel separately. Green means that the signal is on, while gray means that the signal is off. Turn all these buttons to green, so that both channels are turned on.

The field marked 'DGen Ampl' can be used to set the amplitude of the multitone. For this example, we can leave it set to 1.000 FFS, the maximum allowable amplitude.

The digital generator is now set up.

Signal Routing

In order to have something to measure, we need a digital input signal. The signal we will be using in this example will come directly from the digital generator.



Digital I/O Button

Digital signal routing is controlled by the Digital I/O Panel. To show it, select Panels Digital I/O from the menu or press the Digital I/O Button, shown on the left.

You should see a panel that looks like this:

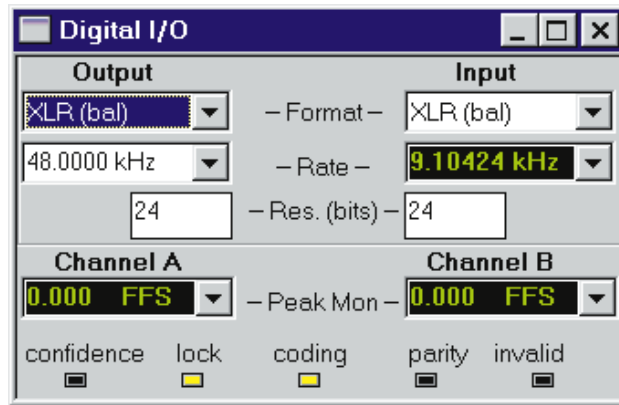


Figure 5-2 Digital I/O Panel

This panel controls the input and output characteristics for digital interface signals. It is also where we can select the internal Generator Monitor path, to route the digital generator directly to the digital analyzer. This will allow us to perform our sample tests without external hardware or cables.

Press the down arrow to the right of the Input Format selector box. From the list, select 'Gen Mon'. This selects the digital generator as the input source to the digital analyzer.

You should see the Peak Monitors on the Digital Analyzer panel change from 0.000 FFS to around 887 mFFS. This indicates that the signal is arriving correctly.

The rest of the settings on this panel control the digital input and output configuration, which are not important since we are using the internal Generator Monitor path. If you were setting up a test on a real device, however, you would want to set these to match your device.

Tutorial - Digital Analyzer Setup



Digital Analyzer
Button

Now that the System Two is set up to generate a digital multitone, and receive it on the input side, we need to set up the analyzer to process the signal. This requires the Digital Analyzer panel. To see the Digital Analyzer panel, select Panels Digital Analyzer from the menu or press the Digital Analyzer Button, shown on the left.

The first thing we need to do is select a program. Hit the down arrow on the selection box and choose 'Multitone generator/analyzer (fasttest)'. The panel should change so that it looks like this:

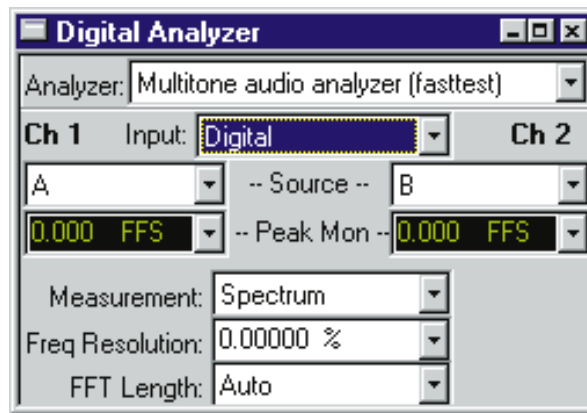


Figure 5-3 Digital Analyzer Panel, Small Version

Press the down arrow to the right of the 'Measurement' box and select the Response measurement mode.

For the response sweep we are trying to create, we don't need to make any other changes to this panel, because the defaults are already correct. We'll take a quick look at the important settings..

Notice that we have selected the Auto FFT Length, which will perform an acquisition that is twice the length of the generator waveform. This is usually the safest choice to make sure the data is valid, but it is not always the fastest. If speed optimization is important, you may want to change this setting.

Since the generation and analysis are both occurring within the same System, a window is not necessary. The acquisition will occur synchronous to the generation. Note that the FFT length must be at

least as long as the generator waveform, which is supported by the choice of 'Auto' for the FFT Length.

The triggering (shown on the large version of the panel) defaults to 'DGen', which is the best choice if the particular trigger time is not critical.

Tutorial - Graph Setup

At this point, we are generating a digital multitone and sending it to the digital analyzer. The digital analyzer is ready to perform an FFT and give us our desired points. All we have left to do is set up the graphing.



To set up graphs, we need the sweep panel, so select Panels Sweep from the menu or press the Sweep Button, shown on the left.

You should see a panel that looks like this:

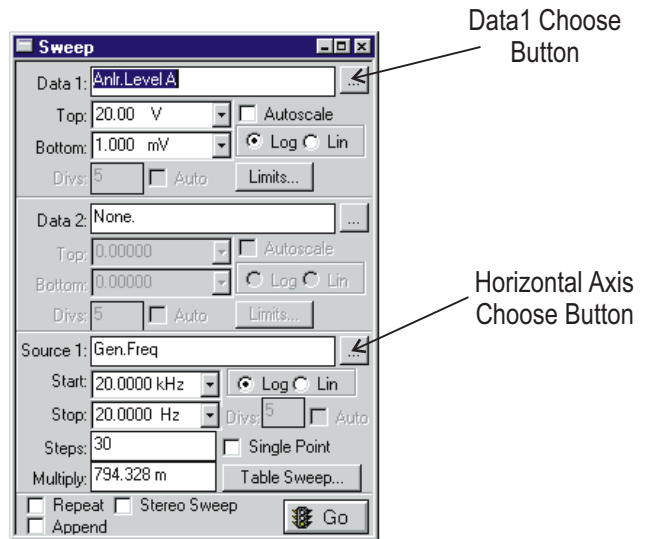


Figure 5-4 Sweep Panel, Small Version

This is the panel that APWIN uses to create a graph or run a sweep.

The first thing we need to change is the horizontal axis parameter. Hit the Horizontal Axis Choose Button (marked on Figure 5-4), and a window will pop up that looks like this:

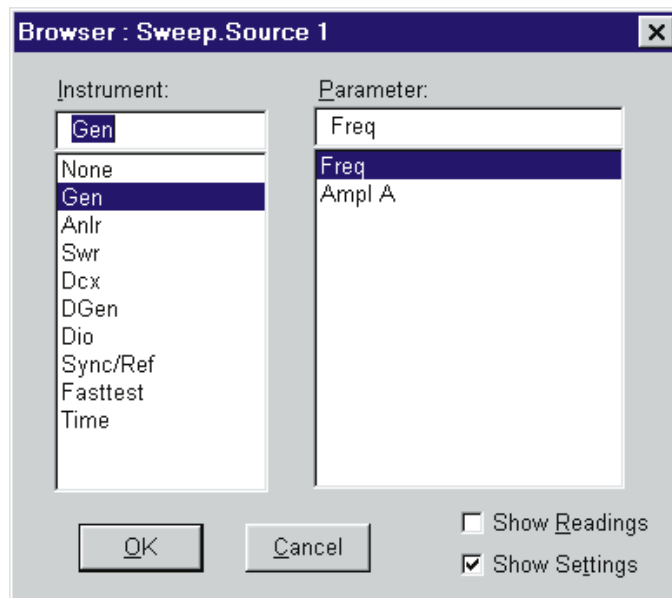


Figure 5-5 Horizontal Axis Browser

This window will allow you to choose a setting from any of the ‘instruments’ within your System One. Choose the Fasttest instrument, and then choose the ‘FFT Freq’ setting. Then hit OK.

Note: The window for choosing the sweep source changes depending on what settings are available. If you don’t see the settings you want, check the Digital Analyzer panel. Make sure the Analyzer is set to ‘Multitone audio analyzer (fasttest)’.

The next thing we need to change is the Data 1 Source. Hit the Data 1 Source Choose Button, and a window will pop up that will allow you to choose the data for the first trace on the graph. Choose the Fasttest instrument, and then choose ‘Ch.1 Ampl’. Press ‘Ok’ and then set the Top of the Graph to 1 FFS and the Bottom of the Graph to 10 nFFS.

Since we want frequency along the bottom of the graph, and we want it to go from 20 Hz to 20 kHz, we don’t need to change the ‘Start’ and ‘Stop’ Fields.

For this application, it is not absolutely necessary that we use a sweep table, provided that we have loaded the generator with the same multitone that we are measuring. However, using a sweep table containing the same points as our multitone will give the most accurate representation of the data. To do this, we use the ‘Table Sweep’ feature of APWIN.

Press the Table Sweep button on the Sweep Panel. You should see a window that looks like this:

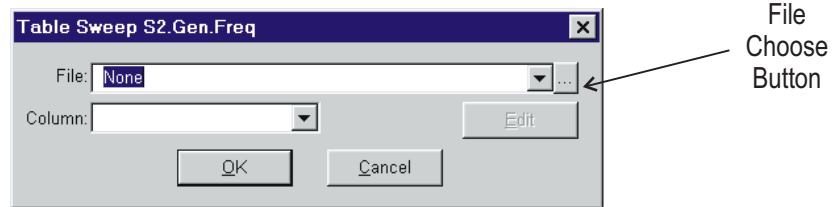


Figure 5-6 Sweep Table Setup Window

This window is where we define the points that will be used for the graph. The points must be taken from a data file somewhere on your hard disk. For this example, we will use ISO31.ads, which is provided with your APWIN software. ISO31.ads is specifically designed to be used with ISO31.agm.

Press the File Choose button (marked on Figure 5-6). You will see a standard window allowing you to choose a data file. Choose the file ISO31.ads, which is located under \Waveform in the directory where you chose to install the sample files. If you used the default locations when installing APWIN, then the full path is C:\Apwin\Waveform\ISO31.ads..

If you like, you may type the full name into the File box instead of pushing the File Choose Button.

Once you have selected the file, the Column Box will read “1=S1.Fastest.FFT Freq”. This does not need to be changed, since the first column of ISO31.ads contains the necessary data for the table sweep.

The Sweep Table Setup Window should now look like this:



Figure 5-7 Sweep Table Setup Window

Now hit Ok to close the Sweep Table Setup Window. When you return to the Sweep panel, you should see that the Left Edge, Right Edge, and Number of Points boxes are gone. This is because the Table Sweep replaces these fields.

Now press the Go/Stop button on the Sweep panel, and you should see a graph that looks like this:

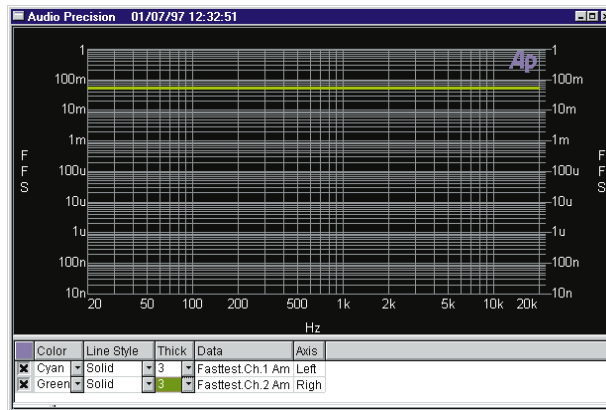


Figure 5-8 Response Graph without DUT

This is the response graph we have been working to create. Since the entire signal path is purely digital, and there is no equalization adjustment, the response is very flat. You can zoom in if you want, but the response will still be a very flat line.

This concludes this exercise in setting up a digital multitone response graph for System Two.

Advanced Concepts

This chapter describes some of the more advanced features used in multitone testing. Certain types of measurements may require these techniques. Other measurements may gain accuracy or speed.

This chapter assumes that you understand the concepts covered in Chapter 2, so if you haven't read it yet, it may be a good idea to look it over.

We will be looking at many of the settings shown on the Digital Analyzer panel. In other chapters, we have left these settings at their default states. In this chapter we will talk about why we may want to change them.

FFT Length

This setting controls how many samples will be recorded during the acquisition.

The advantage of taking more samples is that the FFT creates more frequency bins, and each bin covers a smaller range of frequencies. This means that your FFT will be more able to tell two frequencies apart.

For example, if you are using a 1024-point transform length, and sampling at 48 ks/sec, each bin will be almost 50 Hz wide. So if your multitone contains a 400 Hz tone and a 410 Hz tone, they will fall into the same bin. If they do, there is no way for you or the FFT to tell whether there was one 400 Hz tone, one 410 Hz tone, or one of each. They will all be in the same bin. On the other hand, if the transform length was 16,384 each bin would be about 3 Hz wide, so the two tones would fall into different bins. In this case it would be easy to tell that you had one tone at around 400 Hz and another around 410 Hz.

For another way to look at this, maybe your FFT shows a single spike at 500 Hz. If your transform length is set to 1024, you could only tell that there was a tone somewhere between 450 Hz and 550 Hz. If you had your transform length set to 16,384, you would know that the tone was between 498.5 Hz and 501.5 Hz.

The math involved here is simple. An FFT always measures the frequency range from 0 Hz to half the sample rate. It divides the measurement range into a number of bins (equal to half the transform length plus one), each of equal width. Therefore, if you are running at 48 ks/sec, you will be able to measure frequencies up to 24 kHz. If you are running with a transform length of 8192, then this 24 kHz span will be divided up into 4097 equal bins. Each bin will have the same width in Hertz.

The first bin will be centered on 0 Hz. Half the bin will be above 0 Hz and half will be below. From there, the bins will continue upward in frequency, one after the other. After you have counted off all the bins, you will find that the last bin ends one half bin width below half the sample frequency. The very top half-bin is the 'negative' half of the zero bin.

So, if we use a transform length of 8192 and a 48 ks/sec rate, the bins will be 5.86 Hz wide. The first bin will cover 0 Hz to 2.93 Hz, the second bin will cover from 2.93 Hz to 8.77 Hz, the third bin 8.77 Hz to 14.64 Hz, and so forth. If you wanted to, you could calculate the exact frequency span of every bin.

The disadvantage of a longer transform length is that it takes longer. First, more samples must be acquired, and they arrive at a constant rate, so it will take a little longer. At 48 ks/sec, it takes about 10 ms to obtain 512 samples, while it takes about 340 ms to obtain 16,384 samples. At lower sample rates the difference is even more significant.

In addition to taking longer to acquire, it takes longer to perform a larger FFT.

If you have seen the options for this field, you may be wondering why the available choices are such strange numbers. Why didn't we provide nice, round numbers like 500 or 4000? The real answer to this lies deep within the FFT, but the short answer is that FFT's can be computed much more quickly if they have specific lengths. These lengths are all powers of two - for example, 512 is 2^9 .

Another consideration when selecting the transform length is the length of the generated waveform. The generated waveform has similar restrictions on its frequency resolution. It can only create certain frequencies, depending on the length of its waveform. The frequencies

that the generator can create have the same spacing as the FFT bins, if the sample rate and length are the same.

This means that if you have an FFT length that is twice the generator's waveform length, your FFT will have twice as many bins as your generator could possibly create. If the generator created every frequency it could, it would only fill every other bin in the FFT.

This idea is used in the Noise measurement mode. Not only can the generator only fill the even-numbered bins, distortion products caused by generator signals will also be confined to even-numbered bins. This means that the odd-numbered bins will contain only noise, never distortion or multitone tones. The Noise mode automatically measures only the odd-numbered bins.

When setting up a noise measurement, you want to make sure that the generator waveform length is half the length of the FFT. You can read more about noise measurements and noise mode in the chapter on More Measurements.

If a few seconds doesn't matter to you, you will probably want to select the longest Transform Length possible, to provide the highest frequency resolution. If speed is an issue, you will probably want to use the shortest length you can and still get the resolution you need.

There is also a choice on the panel which will select the Transform Length for you. In System One, it is called 'Maximum', and it will always select the maximum length available to you.

If you choose the 'Auto' Transform Length, the Transform Length will be twice the length of the generator waveform.

Frequency Resolution

The Frequency Resolution field is used whenever the device-under-test frequency-modulates (varies the frequency of) the multitone. The Frequency Resolution should be set to the maximum amount of frequency modulation you expect from the device-under-test.

Many devices cause variations in frequency as the signal passes through them. The most common example of this is the analog cassette player. As a cassette plays, the speed of the tape varies

slightly, which causes small shifts in the frequency of the signal being played.

As you can imagine, these slight frequency shifts can cause the tones to fall into adjacent bins, creating sidebands. This also causes the level of the fundamental frequency bin to be lower.

Without Frequency Resolution, many measurements would become impossible. Frequency Resolution allows the DSP adjust to these variations in frequency.

The Frequency Resolution setting has several effects, depending on the measurement mode.

In Response measurements, errors are created by the lowering of the level of the bins containing fundamental tones. To avoid this problem, the DSP adds up a few bins on either side of each multitone bin, to make sure that all the signal is included. The Frequency Resolution setting tells the DSP how many bins on either side to include.

Distortion measurements normally add up all the bins except the bins containing the multitone tones. Without Frequency Resolution, the distortion calculation may include the sidebands, which will increase the distortion reading. To avoid this problem, the DSP will ignore a few bins on either side of each tone when adding up the distortion. The Frequency Resolution tells the DSP how many bins to ignore.

The Frequency Resolution setting also affects frequency correction. The frequency correction will not correct a frequency error greater than the percentage entered in the Frequency Resolution setting.

Triggering

The Triggering settings (Triggering and Trigger Delay) tell the DSP when to begin the acquisition. Since acquisition times can be very short, it can be critical to the measurement that the acquisition start at the correct time, so that the correct signal is measured.

You must always press Go before the acquisition cycle will begin - before you press Go, System Two will not even be watching for trigger events. After you press Go, the selected event will begin the acquisition.

FASTTEST for System Two allows four different types of trigger events:

1. DGen Trigger
2. Signal Trigger - Tight, Normal or Loose
3. External Trigger
4. Off

Off is the simplest option, and this is selected by default. If Off is selected, the DSP doesn't wait for anything. As soon as you press Go, the acquisition begins.

DGen Trigger is similar to Off in that it doesn't require any external signal. Once you press Go, the DSP waits for the digital generator to start its cycle, and begins the acquisition in synchronization with the digital generator.

External Trigger allows you to provide an electronic signal to tell the DSP to begin its acquisition. The signal must be provided to the General Purpose Serial I/O Connector, on pin 3. If the pin is held low (GND), the acquisition will wait. As soon as the pin goes high (+5V), the acquisition will begin. Allowing the pin to 'float', or not connecting it, is the same as the pin going high.

Signal Trigger (Tight, Normal, or Loose) is the most complex and most powerful triggering source. When you are in signal trigger mode, the DSP will monitor the inputs and trigger when it sees a certain signal. Usually you will want it to trigger when it sees your multitone.

In order to trigger on a certain signal, the signal must be loaded into the DSP's generator buffer. The generator buffer is a memory area where the DSP stores the arbitrary waveform to generate.

If the DSP is currently generating a multitone, then that multitone is already stored in the generator buffer. If you are using something else to generate the multitone, then you will need to load the multitone into the generator buffer. To do this, select the 'Arbitrary Waveform' selection on the Digital Generator or Digital Analyzer panel and select a file describing your multitone. This will automatically send the waveform data to the DSP, and it will be ready to detect that signal. If you wish, you may then select a different waveform type (as long as

you don't select another arbitrary waveform), and the generator buffer will remain loaded.

Once you press Go, the DSP will continuously monitor the incoming signal and compare it to the waveform in the generator buffer. When it finds that they match, the DSP will trigger and begin acquiring.

The two waveforms don't need to exactly match, but they must be close. The DSP basically does an FFT of each and compares the two. Any bins which have a main tone in one FFT must have a main tone in the other FFT. Any bins which have low signal amplitude in one FFT (distortion or noise) must have also a low signal amplitude in the other.

The choice of Tight, Normal or Loose tells the DSP how much difference to tolerate. If you choose Loose, the DSP will trigger on signals quite a bit different from the generator buffer, which means that you may accidentally trigger on a signal that is not your desired multitone. If you choose Tight, the DSP will only trigger on a signal that is virtually identical to your multitone. This reduces the chances of an erroneous trigger, but if your device-under-test modifies the signal very much (by altering the frequency response or adding distortion) the DSP not trigger at all.

The Frequency Resolution setting also affects the triggering. If your device-under-test alters the frequency of the multitone, you will want to set the Frequency Resolution to the maximum amount of frequency deviation that you expect. This tells the DSP to check for frequency variations, and to trigger if it finds a frequency-altered signal that matches the trigger criteria.

In general, you should choose longer generator waveform lengths when using signal triggering. If you use waveforms shorter than 2048 samples, the small number of bins makes it difficult for the triggering to work reliably, resulting in false triggers. If you use a short record length, using Tight triggering criteria will reduce the chances of false triggers.

The choice of trigger criteria depends a lot on your particular testing circumstances. In general, you probably want to use the tightest trigger criteria that will reliably trigger. Start with Tight, and if it won't trigger (because your device affects the signal too much), change to Normal.

If it still won't trigger, change to Loose. If you get false triggers, you will want to go up to Normal or Tight. If you have trouble finding a criteria that triggers reliably and yet rejects false triggers, you should consider using a longer waveform.

How long is the waveform you are using? APWIN won't tell you, and neither will the DSP, so you just have to know. When you create your own waveforms (using MAKEWAVE - see Chapter), it is a good idea to keep a list of the filenames and all the important data, including the length of the waveform.

You may use synchronous processing techniques to determine the length of the waveform. See the section on Synchronous Processing, page 6-12.

Trigger Delay

Trigger Delay causes the System to wait for a period of time between the trigger event and the start of acquisition.

Since acquisition times are limited, trigger delay may be used to capture the signal you want to observe, even if it doesn't happen directly after the trigger event.

Trigger delay can also be used to give a device-under-test time to adjust to a new signal. Many devices are not ready to be measured the instant the signal is applied - they must be allowed to process the signal.

Trigger Delay may be used with any of the Trigger Sources. A delay time may be entered for the Trigger Delay, and the DSP will wait for this period of time, after detecting the trigger condition, before starting the acquisition.

Note that if you are using only a short burst of the multitone to trigger and acquire on, the minimum burst length required is increased by the trigger delay time.

FFT's: The Looping Problem

This section describes why windows or synchronous processing techniques are necessary when using FFT's. If you are comfortable with the use of windows and the reasons why we use them, you may want to skip forward to the discussion on synchronous processing.

Before performing an FFT, we acquire a buffer full of sample data. The mathematics of an FFT require that the sample data form a continuous loop. When the FFT is performed, it assumes that the sample data is not a single 'slice' of time, but rather one cycle of a continuing loop.

For example, let's say we sample this single sine wave:

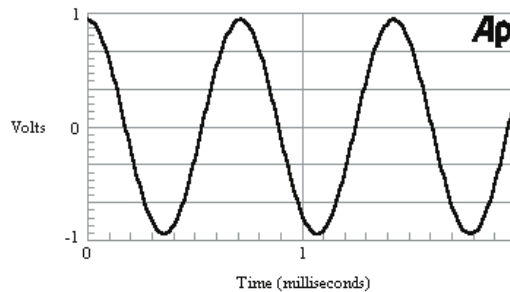


Figure 6-1 Single Sine Wave

Since this is only a single sine wave, we would expect that if we performed an FFT on it, we would see only a single spike, at the frequency of the sine wave (1.4 kHz).

But when the FFT is computing, it sees the sine wave like this:

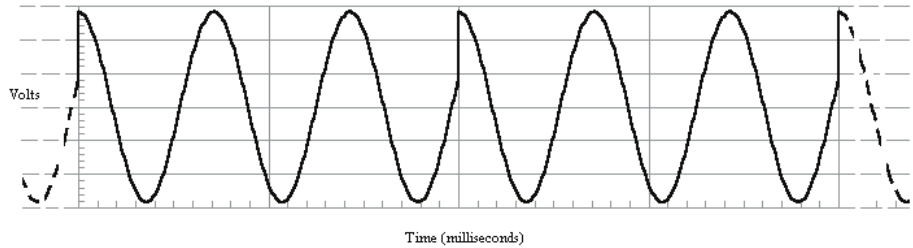


Figure 6-3 Single Sine Wave As Seen by FFT

The dotted lines at the beginning and the end of the sample are intended to show that the FFT thinks that the signal continues on forever in both directions. No matter what signal you sample, the FFT will assume that it is one cycle of an endless loop.

So, as far as the FFT is concerned, we don't have a simple sine wave. We have a single sine wave with a big jump in the middle of it. This jump looks like a lot of frequencies to an FFT. Instead of seeing only a single spike in our FFT, we get a graph like this:

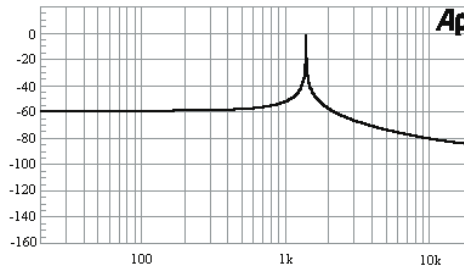


Figure 6-2 FFT of Sine Wave with No Window

Compare this result to Figure 6-6, which shows the graph that can be obtained using a window. Notice the difference in the noise floor. If we don't use a window, the jump in the waveform causes us to lose a lot of the low-amplitude information.

Windows

One way to make our FFT data more useful is to apply a ‘window’ to the sample buffer. A window just shrinks the signal at each end of the sample buffer so that the buffer begins and ends at zero. Therefore there is no jump in the looping waveform.

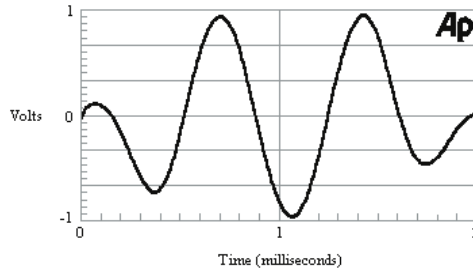


Figure 6-4 Sine Wave with Window

Figure 6-4 shows a sine wave with a window applied to it. The window tapers the sample buffer at both ends so that it begins and ends with zero. This avoids the problem of having a jump in the middle of the waveform. To the FFT, the waveform will now look like this:

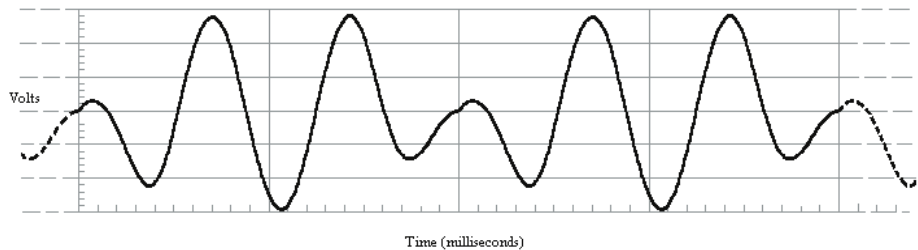


Figure 6-5 Sine Wave With Window, As Seen by FFT

Notice that the waveform connects smoothly. There is no longer a jump in the waveform. However, the signal is still somewhat distorted. It's still not exactly like the simple sine wave that was sampled, but it is better.

Using a window, we can get an FFT that looks like this:

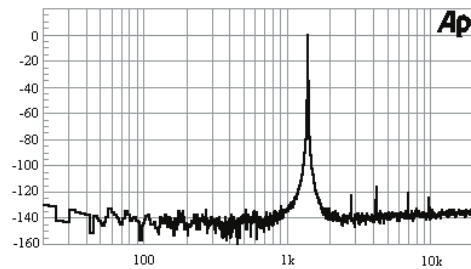


Figure 6-6 FFT Using Hann Window

Notice how much more detail we can see, compared to Figure -2. We can see the noise floor a lot better, and we can even see the harmonic distortion. This is definitely a more useful graph.

We still don't see a lone spike, though. The 1.4 kHz fundamental tone still spreads out at the base. This is due to the window.

There are several different types of windows. All of them taper the waveform at both ends, but the tapering is different for each window. Some taper more rapidly than others, and each window has its own characteristic shape for the tapering. The windows you will see in System One or System Two are Hann, Blackman-Harris, Flat-Top, and Equiripple (but only Hann is available to *FASTTEST*).

Each window will have a unique effect on the frequency graph, depending on the shape of the windowing. No window is perfect.

Synchronous Processing

Synchronous processing allows us to perform FFT's without using a window, but without causing jumps in the waveform.

The first ingredient we need is a repetitive waveform. Multitone waveforms are repetitive, since they are generated from a list of samples that is repeated in an endless loop.

These samples, which come from a file created by MAKEWAVE, are designed so that they are continuous. When constructing the waveform file, MAKEWAVE will make sure that there is no jump in the waveform as it loops from the end back around to the beginning. It does this by carefully calculating each sample so that each time the generator reaches the end of the samples and starts again at the beginning, it picks up right where it left off.

For example, let's say the multitone file contains this waveform:

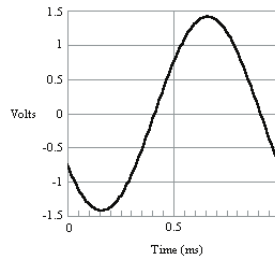


Figure 6-7 Single Cycle of Generator Waveform

Then, when the waveform is repeated, it will look like this:

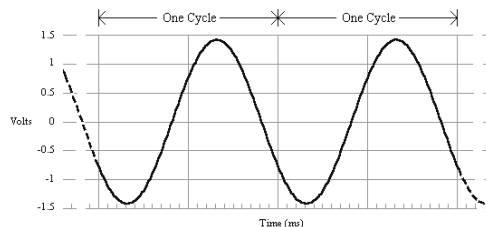


Figure 6-8 Looped Generator Waveform

Notice how each cycle blends seamlessly into the next so that there are no jumps. Most multitone waveforms will contain more than one sine wave, but each and every sine wave must blend smoothly during the transition from one cycle to the next.

If you have used MAKEWAVE to construct multitone waveforms, you probably noticed that only certain frequencies are available. This is exactly why - they must be loopable at the length of the sample file.

Any frequency to be included in the sample file must be repeated a whole number of times.

Synchronous processing uses this same idea. If we capture exactly one cycle of the generator waveform, it will form a perfect loop and we can perform the FFT without using a window. We may capture more than one cycle, but we must capture whole cycles. If we capture part of a cycle, the FFT will see a jump in the waveform.

It doesn't really matter whether we start at the beginning of the generator cycle or somewhere in the middle. As long as we capture whole cycles, they will be continuous and we can do the FFT without a window.

The 'Processing' selection on the Digital Analyzer panel controls windowing and synchronous processing. There are three choices: Synchronous, Freq Corrected and Windowed.

Synchronous is used for synchronous processing, and Windowed uses a Hann window (for situations for a synchronous signal is not available). We will talk about Freq Corrected in a minute.

True synchronous processing does not require any special features of the DSP. As long as the setup is such that acquisition will capture exactly one or more complete cycles of the generated waveform, and no window is applied, synchronous processing will occur.

This requires the following:

- The signal must originate in the same System Two at the same time as the acquisition. It cannot come from a different System or be prerecorded and reproduced by the device-under-test.
- The device-under-test must not change the frequency of the audio signal.
- If a digital device, the device-under-test must not change the sample rate.
- The acquisition buffer must be as long as, or longer than, the generator waveform, and must be an even multiple of the generator waveform length.

If you are not certain how long your generator waveform is, you can use synchronous processing with the longest available acquisition. If you want, you can then reduce the acquire length step-by-step, taking an FFT each time, until the FFT shows a noticeable change. When your acquire length is shorter than the generator waveform, you should see a graph similar to Figure 6-2. Depending on the multitone you are using, it may have different peaks in it, but you will notice a significant rise in the noise floor and spreading around each peak.

If you are using a different System for generating than for analyzing, or if you are analyzing a multitone that is being generated another way, you cannot use true synchronous processing. Even if the acquisition is set for the same length as the waveform cycle, the slight differences in sample rate may cause the acquisition to be just a little more or less than the waveform cycle. Even a difference of one sample will cause a slight jump in the waveform and a significant increase in the noise floor.

These situations require use of 'Freq Corrected' processing. This type of processing checks the acquired signal, before performing the FFT, and mathematically adjusts it to correct the difference in timing. It can correct frequency shifts of up to 3%.

You would want to use this option in any situation where the signal is not being simultaneously generated and analyzed by the same System, or where the device-under-test intentionally changes the frequency of the signal as it passes through.


The main disadvantage of frequency corrected processing is that it takes more time. Not a lot more, but if speed is your primary consideration then another option should be considered. Another possible disadvantage is that your FFT will show the corrected frequencies, not the actual frequencies being output by your device. If you want to know exactly how much your device is altering the frequency, then you should not use frequency correction.

Multiple Measurements on the Same Data

Sometimes it is desirable to make several complete measurements from a single acquisition cycle. The main advantage to this is in speed; the acquisition of the digital audio data can be the slowest part of the testing process. Another advantage is that the device-under-test is only required to be carrying the signal during the acquisition cycle, and the data processing can happen later.

Multiple Measurement Concepts

If you are not familiar with the sequence of events during an FFT, it might be worthwhile to review the 'Sequence of Events' section of Chapter 2.

A complete FFT measurement occurs in four distinct steps, starting when you press 'Go' or hit  :



1. Wait for trigger event
2. Acquisition (recording digital data)
3. Perform FFT (transform)
4. APWIN requests points and graphs data




As APWIN requests data from the FFT, the DSP performs some post-processing computations on the FFT data, and passes the results of these computations back to APWIN. The computations that *FASTTEST* perform vary depending on the measurement mode of the DSP. For example, in Noise measurement mode, the value that the DSP passes back to APWIN is the sum of every alternate bin between the last frequency requested and the current frequency requested.



None of the calculations that are performed by the DSP change the data. They only affect the value sent back to APWIN, so both the acquisition data and the FFT data remain in the DSP.

This means that the DSP can perform other computations on the data. If you like, you can even change the measurement mode and get an entirely different measurement from the same data. You can measure a different parameter or change the points that you request. If it is

planned carefully, you can get almost any combination of measurements from a single acquisition.

The APWIN feature that is required is called 'Reprocess Data' and is available from the Sweep menu, or by holding down  and pressing . This causes the APWIN to request all the points again from the DSP without reacquiring the data or recalculating the FFT from the raw data. If there is a different measurement mode in effect, the DSP's post-processing calculations will be different. If there is a new set of points in effect, APWIN will request these new points instead of the old ones.

So, if you want to take several different measurements from the same set of data you begin by setting up your first measurement and running a normal trigger-acquire-transform-graph cycle by pressing , by pressing the 'Go' button on the sweep panel or the main toolbar, or by selecting Sweep Start from the main menu. Then you can change the settings to obtain another measurement, and press   to get the next measurement from the same acquired data.

Another way to get multiple test results from a single acquisition is to run the first measurement, then load a new Test with different settings, and then press   to obtain the new measurement from the old data.

Whether you are loading a new Test or making changes on the panel, you must be careful what settings change as a result. If certain settings change, the data may be destroyed.

The following settings may be changed without destroying the data:

- Measurement mode
- Freq Resolution
- Phase Display mode
- Any settings on the Sweep panel
- Triggering and Trigger delay settings (but these will have no effect since the data will not be reacquired).

These settings must **NOT** change, or the data may be destroyed:

- Analyzer
- FFT Length
- Generator waveform
- Sample Rate

Automating Multiple Measurements

Multiple measurements on the same data set can be automated, but this requires the use of Procedures. A Procedure is basically a list of commands for APWIN to follow, which it follows just as if you had entered the commands using the keyboard or mouse. The commands are written in a simple computer language called APWIN BASIC.

The easiest way to create a Procedure is to let APWIN do it for you. In APWIN versions 1.50 and later, there is a Learn feature which automatically follows your keystrokes and mouse moves and writes a Procedure for you. Then, when you play back the Procedure, APWIN will repeat the same process you went through while you had Learn Mode on.

You can also write Procedures ‘from scratch’, by typing the commands into the Procedure editor instead of using Learn Mode. Or you can do both - use Learn Mode to write the Procedure for you and then make changes to the Procedure using the Procedure editor.

Procedures have a lot of power built into them, allowing you to automate any process that APWIN can do and also to interact with other programs on your computer, such as a word processor or spreadsheet. A complete discussion of how to use all the features of the Procedure language is available in the “APWIN BASIC Users Guide and Programming Reference”. A simpler tutorial is given in “Getting Started with APWIN BASIC Procedures.”

However, we can demonstrate multiple measurement capabilities with minimal knowledge of Procedures, and show you how to set up your own multiple measurement suites.

We will show an example of using Learn Mode creating a Procedure that uses one acquisition cycle to give 31-point graphs of frequency response, noise, and phase.



New Test Button

Multiple Measurement Tutorial

The first thing we need to do is to set up the first multitone measurement. Start APWIN and reset to a New Test by pressing the New Test Button, shown on the left. Put up the panels shown on the following four figures and set them so that they appear as shown:

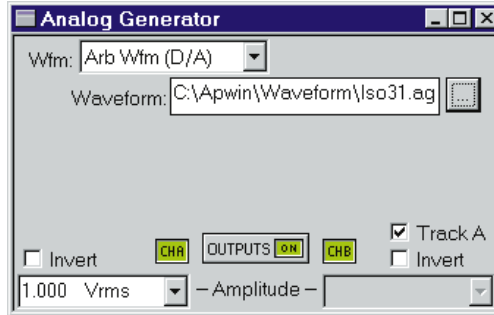


Figure 7-1 Analog Generator Panel for System Two

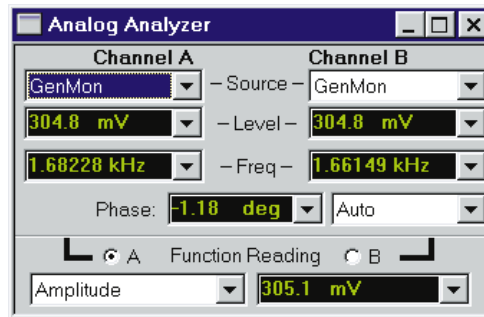


Figure 7-2 Analog Analyzer Panel for System Two

Some settings affect the available choices in the panel areas below them, so if there is a setting that you can't match to the picture, make sure the settings above it (on the same panel) are correct.

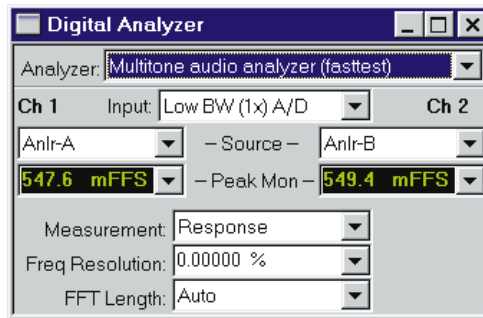


Figure 7-3 Digital Analyzer Panel for System Two

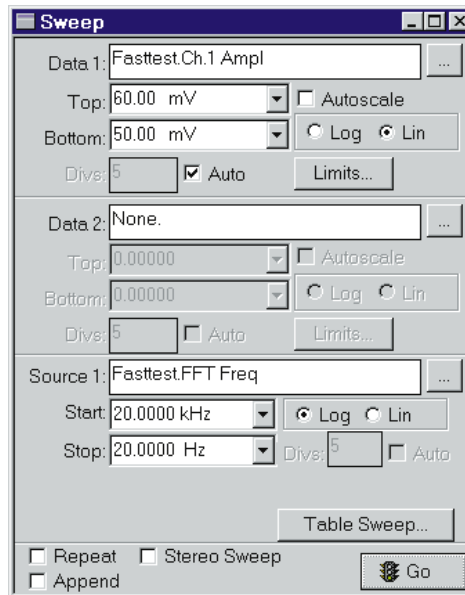


Figure 7-4 Sweep Table Set Up for Response Test

To set up the Generator panel, you must select 'Arb Wfm (D/A)' in the 'Wfm:' box before you can choose the filename. Then press the Choose Waveform box (marked on Figure 7-1) and use the file browser to find the file Iso31.agm. If you installed using the default directory structure, it will be in C:\APWIN\Waveform.

The Sweep panel must be set up to use a Sweep Table. To do this, press the Table Sweep button. You will see a window that looks like this:

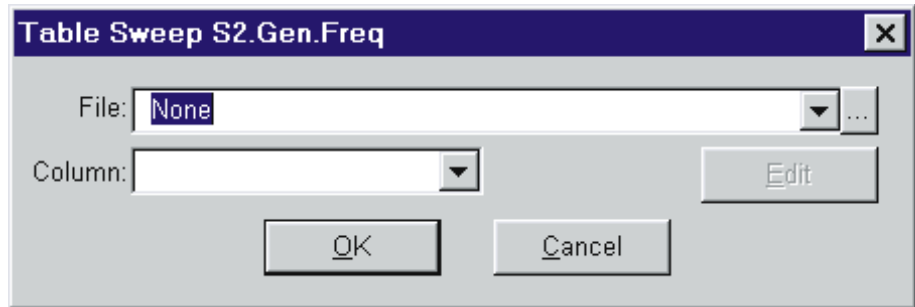


Figure 7-5 Default Sweep Table Setup Window


Press the browse button (marked on Figure 7-5) and use the browser to find and choose 'Iso31.ads'. If you installed your samples into the default directories, this file will be in the folder 'C:\APWIN\Waveform'. If you chose a different directory to install your samples, the exact path may be different.

After you have chosen the correct Sweep Table file, you must choose the column from which to take the data. The default, column 1, is correct in this case. The panel should now look like this (although the path to Iso31.ads might be different):



Figure 7-6 Sweep Table Setup Window Set Up For Response Test

Press OK to dismiss the Sweep Table panel, accepting your choices.

Start the sweep by pressing 'Go' on the sweep panel, or selecting Sweep Start from the menu, or by pressing . You should see a graph that looks like this:

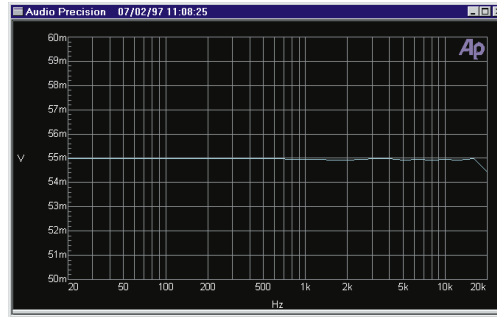



Figure 7-7 Frequency Response of System Two

This graph shows System Two's overall frequency response from the D/A's, through the Analog Generator, the Analog Analyzer, and the A/D's.

Use File Save As Test to save this panel configuration as a test file. Name it 'Response'.

Change the Measurement mode on the Digital Analyzer panel to Noise. On the Sweep panel, change the Top field for Data1 to 50 uV and the Bottom field for Data1 to 0 V.

Use File Save As Test to save this panel configuration as a test file. Name the Test 'Noise'.

Press  or select Sweep Reprocess Data from the menu. You should see a graph that looks like this:

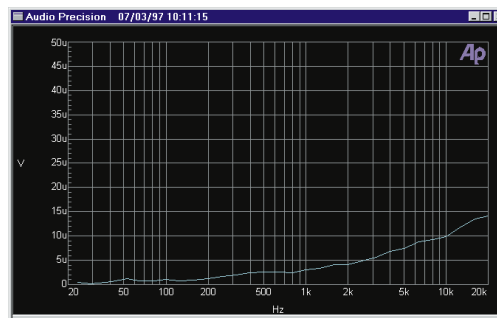


Figure 7-8 System Two Noise vs. Frequency

This graph shows System Two's frequency response over the entire signal path. This graph uses the same acquired data that was used by the previous measurement.

Now expand the Digital Analyzer panel and set 'Ch. 2 Phase Display' to 'Interchannel' and the Measurement mode back to 'Spectrum'. On the Sweep panel, change the Data1 Source to 'Fasttest.Ch.2 Phase'. Change the Data1 Top to 2 deg. Change the Data1 Bottom to -2 deg. The Sweep panel should now look like this:

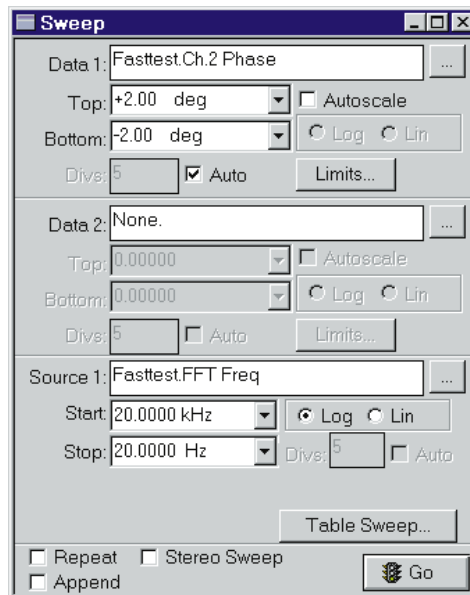



Figure 7-9 Sweep Panel Set Up for Phase Measurement

Use **F**ile **S**ave **A**s **T**est to save this panel configuration as a test file. Name the Test 'Phase'.

Press  or select Sweep Reprocess Data from the menu. You should see a graph that looks like this:

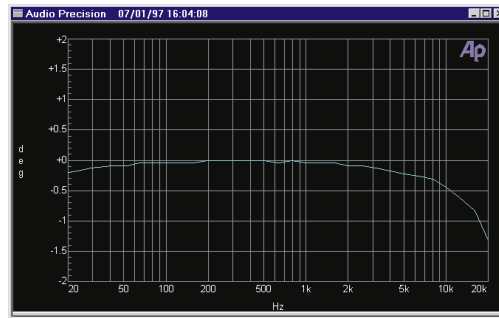



Figure 7-10 System Two Phase vs. Frequency


This graph shows System Two's phase shift over the entire signal path. Again, we are using the same data originally acquired.

So far, we have created 31-point graphs of the Response, Noise, and Phase of the System, all from a single acquisition, and saved each configuration as a Test file. Now we will combine them into a Procedure that will run all three tests automatically.

First select Utilities Learn Mode from the menu. Your mouse pointer will change and a new window called "Procedure1" will appear. From now on, everything you do in APWIN will be automatically written as APWIN BASIC instructions in this window. At first the Procedure window will be in front of APWIN, but as soon as you click on APWIN it will come up to the front.

Be careful not to make any extra movements - every menu item or button you select will automatically be recorded into the Procedure.

Open the Response test that you just saved, by selecting File Open Test from the menu and selecting the filename from the browser. Run a full acquisition/transform/post-process/graph cycle by pressing 'Go' on the sweep panel, or selecting Sweep Start from the menu, or by pressing .

Load the Noise test the same way you loaded the Response test, then press  or select Sweep Reprocess Data from the menu.

Load the Phase test, and press  or select Sweep Reprocess Data from the menu.

Now select Utilities Learn Mode from the menu. Your mouse pointer will change back to a normal arrow. Your actions will no longer be recorded in the Procedure.

Now take a look at the Procedure window. In order to see it, you may have to minimize APWIN or select the Procedure Editor from the taskbar. It should look like this:

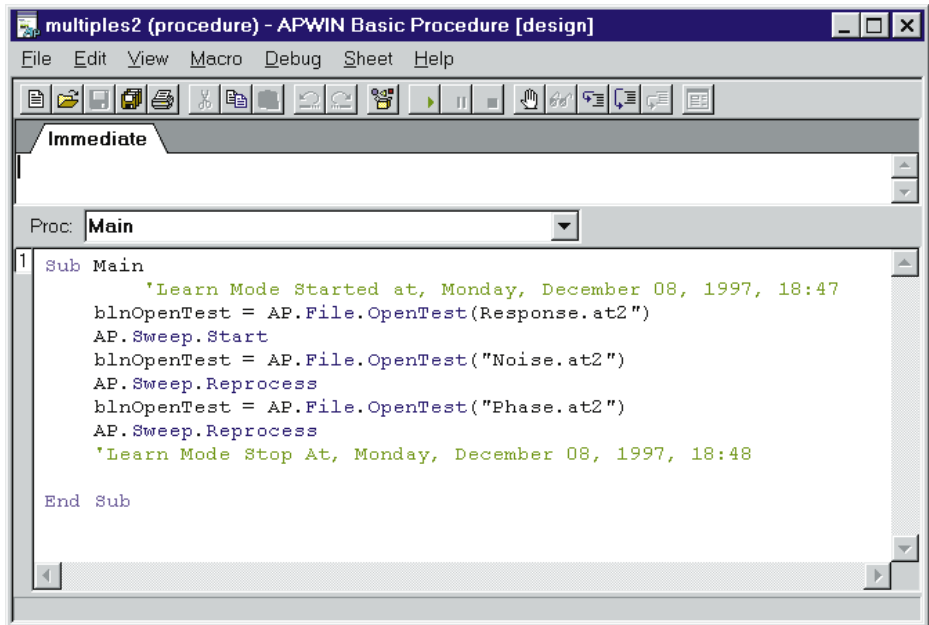


Figure 7-11 Procedure Editor After Learn

These are the instructions to load and run the three tests from one acquisition. The wording of APWIN BASIC makes it fairly apparent what command is being performed, but if you want more detail, consult the “APWIN BASIC Users Guide and Programming Reference”. Run the Procedure by pressing the button shown on the left, or by selecting Procedure Run from the main APWIN menu.

This Procedure is not particularly useful, since it probably goes too quickly for you to see the data. However, you can add limits to your

tests and generate a report of any failures. Or you can add printing of the graphs to your Procedure (using Learn mode to write the graph-printing commands for you) and get a printout of each of your results.

If you like, you can save the Procedure from either APWIN or the Procedure Editor. Once you save the Procedure, it can be reloaded and run at any time. Next time you run it, however, make sure that the referenced Test files are accessible, in the same folder where the Procedure 'learned' them. The Procedure will need to find the Test files in order to run.

To save the Procedure from APWIN, select File Save As Procedure and choose a filename.

To save the Procedure from the Procedure Editor, select File Save As and choose a filename.

To reload the Procedure at a later time, select File Open Procedure from the APWIN menu, and use the file browser to choose the Procedure file. Then you can run the Procedure by pressing the button shown on the left, or by selecting Procedure Run from the main APWIN menu.

If you want to start over, and learn a new Procedure, go to the Procedure Editor and select File New from the menu. This will start you with a fresh Procedure. For more detail on how to use Procedures and the Procedure Editor, see the other manuals that came with APWIN.

Generating Custom Waveform Files

Sometimes it may be desirable to create your own custom multitone. You may want to do this so that you can choose the data points you desire, so that you can make a particular combination of measurements, or so that your multitone selection accurately represents the spectral content of your program material.

There are a number of multitone waveform files provided with APWIN. Before creating your own custom multitone waveform file, you may want to check to see if any of the samples suit your needs.

Custom multitones are created using a program called Makewav2. This program is automatically installed when you install APWIN, and an icon is created in your Audio Precision APWIN application group.

There are a number of options to Makewav2 that must be considered when creating your own waveform file.

Sample Rate

All waveforms generated by Makewav2 are created for a certain sample rate. While they can be used at other rates, all the frequency components will be scaled accordingly. For example, if you create a waveform with a 1 kHz tone at 48k samples/sec, and then run the generator at 44.1k samples/sec, the tone will be 918.75 Hz. It is best to create a waveform for only one sample rate, and commonly the sample rate will be part of the file name.

Makewav2 also generates the data that is used to create the sweep table. When you load a sweep table into APWIN, the frequencies in the sweep table are absolute, regardless of the sample rate. Therefore, if you change the sample rate of the generated waveform, the tone frequencies will no longer match the sweep table and you may get incorrect data.

Waveform Length

FASTTEST, *FASTTRIG* and *Codec* can use waveforms that are 8192, 4096, 2048, 1024, 512 or 256 samples long.

The main considerations are speed and frequency resolution. The choice of waveform length is usually a compromise between these two factors. If you want the fastest possible operation, you should choose the shortest length. If you want the best frequency resolution, you should choose the longest length.

The speed differences exist on the analysis side of the process. While it doesn't take any longer to generate a longer waveform, it generally means that a longer acquisition length should be used. A longer acquisition takes longer to acquire and longer to process the FFT.

The most significant speed impact is felt when using Signal Triggering. In this type of triggering, the acquisition begins when the DSP recognizes your multitone. It takes the DSP a certain amount of time to recognize the signal. Therefore, there is always a minimum burst length that the DSP requires in order to trigger and acquire, and this burst length depends on the length of the waveform. The following table shows the approximate minimum burst length required for each waveform length, at the most common sample rates:

Waveform Length (samples)	Sample Rate (samples per second)				
	24k	32k	44.1k	48k	54k
8192	555 ms	840 ms	1020 ms	1110 ms	1250 ms
4096	275 ms	420 ms	510 ms	550 ms	620 ms
2048	140 ms	210 ms	260 ms	280 ms	320 ms
1024	75 ms	115 ms	140 ms	150 ms	170 ms
512	40 ms	60 ms	75 ms	80 ms	90 ms
256	25 ms	40 ms	46 ms	50 ms	60 ms

Table 10-1: Minimum Burst Length

The frequency resolution is the spacing of frequencies that can be created by the waveform. In any digital generator situation, there are only certain frequencies that can be generated in a constant loop. The longer the loop, the more frequencies can be generated, and the closer the available frequencies are spaced. This also affects the lowest frequency that can be generated, which is the same as the minimum spacing. The formula for the minimum spacing is the sample rate divided by the waveform length in samples. The following table shows the approximate minimum spacing for each waveform length at the most common sample rates.

Waveform Length (samples)	Sample Rate (samples per second)				
	24k	32k	44.1k	48k	54k
8192	2.93 Hz	3.91 Hz	5.38 Hz	5.86 Hz	6.59 Hz
4096	5.86 Hz	7.81 Hz	10.77 Hz	11.72 Hz	13.18 Hz
2048	11.72 Hz	15.63 Hz	21.53 Hz	23.44 Hz	26.37 Hz
1024	23.44 Hz	31.25 Hz	43.07 Hz	46.88 Hz	52.73 Hz
512	46.88 Hz	62.50 Hz	86.13 Hz	93.75 Hz	105.5 Hz
256	93.75 Hz	125.0 Hz	172.3 Hz	187.5 Hz	210.9 Hz

Table 10-2 Minimum Tone Spacing

Since the acquire length is generally related to the waveform length, the bin widths in your FFT should also be considered. Bin widths are important for any measurement that requires frequency accuracy. For more discussion on the acquire length and bin width, see Chapter 9’s discussion of FFT Length.

Frequency Considerations

When you create a multitone waveform, you must choose the frequency, phase, and amplitude of each frequency component. Since only certain frequencies can be created in a continuous loop, MAKEWAV2 will round the requested frequencies to the nearest loopable frequency.

The components that you choose depend on the measurements you intend to make. If you want a lot of points in your graph, you will want a lot of frequency components.

When choosing the frequency components, it may be important to consider distortion products. If the device-under-test creates distortion, and the frequency of the distortion is the same as one of the points in your multitone, the distortion product will affect the value in that bin. This may affect the quality of a response measurement at that data point.

If you are measuring distortion, and a distortion product falls into the same bin as one of the fundamental tones, it will be impossible to measure the distortion at that point.

Distortion products are always small whole-number multiples of the 'fundamental', the frequency causing the distortion. These small whole-number multiples of the fundamental are called 'harmonics'. For example, if you have a 200 Hz tone in your multitone, it will create harmonics at 400 Hz, 600 Hz, 800 Hz, and so forth. The largest distortion products are usually the lower-frequency ones, although depending on the nature of the distortion, it may emphasize even harmonics, odd harmonics, or even just a single harmonic.

Distortion products may be created at harmonics of each of the frequency components in your multitone, so you probably want to place all your tones so that they are not on the harmonics of any of the other tones.

If the device you are testing creates a significant amount of intermodulation distortion, you will also want to avoid placing tones at frequencies that coincide with the sum and difference products of other tones.

Amplitude Considerations

When you specify each frequency component, you can also specify its amplitude relative to the other frequency components. You can use linear units, such as Volts, or logarithmic units such as dBV. The exact values that you request are ignored; only relative differences between the components are significant (unless the Absolute Amplitude feature is used).

Waveform files are most commonly created with all frequency components at equal amplitudes. For most measurements, this arrangement is most convenient. In some circumstances, however, you may want to create components at varying relative amplitudes. Usually, this will be because you want to duplicate the general shape of your program material, or to offset frequency deviations within the device-under-test, resulting in a flat output. It may also be useful in testing pre-emphasis or de-emphasis characteristics of a device.

When constructing the wave file, Makewav2 will normally compute the signal peak and scale the components so that the peak does not exceed digital full scale (the maximum allowable value). If digital full scale is exceeded, distortion will result.

Makewav2's computations of the peak value are limited in accuracy, so the waveform is not scaled to exactly touch digital full scale, but to reach a value a certain number of dB below full scale. This decibel value is called the headroom, and allows for some error in the computation of the peak value. The headroom defaults to 1 dB, but can be changed. If you set the headroom smaller than 1 dB, you increase the risk of the signal reaching full scale and causing distortion. If you set the headroom greater than 1 dB, you reduce the risk of hitting digital full scale, but you will decrease the signal-to-noise ratio of the signal.

Rather than specifying the headroom, in some situations it is desirable to specify the absolute amplitude of your frequency components. This is usually useful when creating stereo waveforms with different frequency components and/or phases in each channel, but where equal amplitude frequency components are desired. Because different frequencies and phases are used, the peak values of the two waveforms will probably be different, so the waveforms will be scaled

differently. This means that a 0 dB component in one channel will be at a slightly different amplitude than a 0 dB component in the other channel, and there will appear to be a stereo imbalance.

If you wish to specify the absolute amplitude of your components, you must specify the amplitude of each component as 1 Volt or 0 dB. Then, when you run Makewav2, you invoke the Absolute Amplitude option and specify the amplitude, in dBFS, of a 1 Volt or 0 dB component.

If you use the Absolute Amplitude option, the headroom parameter is ignored, since Makewav2 is no longer in control of scaling the amplitude. This means that unless you are careful, you may specify an absolute amplitude that causes the waveform to reach above digital full scale. Usually the best way to prevent this is to run Makewav2 on the multitone first without specifying the absolute amplitude. The report that Makewav2 gives will tell you the absolute amplitude, and this data will tell you the maximum absolute amplitude that stays within the specified headroom. Then you can rerun Makewav2 and specify the absolute amplitude, keeping it at or below the amplitude given by the first run. You will then be assured that the headroom is sufficient.

For example, say you were creating a stereo waveform, with different components in each channel, but you wanted them to have equal amplitudes. First, you create the left waveform and run Makewav2 on it, allowing 1 dB of headroom. Makewav2 reports that the absolute level produced is -20.43 dBFS. Next, you create the right waveform and run Makewav2 on it, allowing 1 dB headroom, and it reports that the absolute level is -21.29 dBFS. Then you will want to recreate both waveforms, specifying an absolute amplitude of -21.29 dBFS for both of them. Then the components of both waveforms will have equal amplitudes.

Phase Considerations

The phase of each frequency component in your multitone may also be specified. If you don't specify the phase, all components will be constructed at 0 degrees. Usually this is the best choice.

The phase of a frequency component determines where the sine wave starts at the beginning of the generator loop. For example, here are two frequency components at different phases:

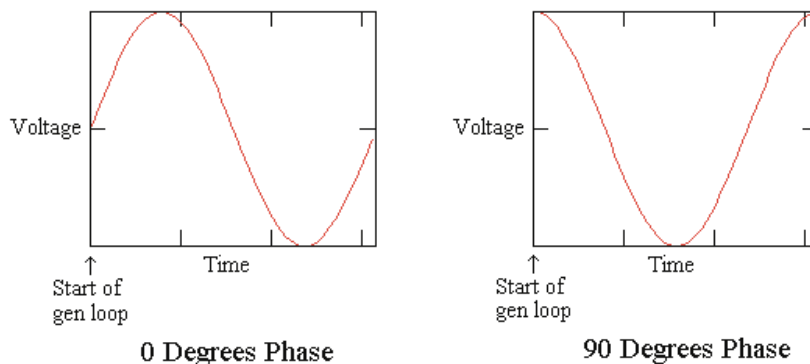


Figure 8-12 Two Sinewaves With Different Phases

Notice that the 0 degree wave starts at zero amplitude, while the 90 degree wave starts at maximum amplitude. If your multitone only uses one tone, the phase makes no difference whatsoever, since the generator runs in a constant loop. However, when you have several tones, it begins to make a difference.

The next example shows two multitone waveforms, each with four tones. One has all the tones starting at 0 degrees, the other has all the tones starting at 90 degrees. The dashed lines are the sinewave tones, and the solid lines are the resultant waveforms.

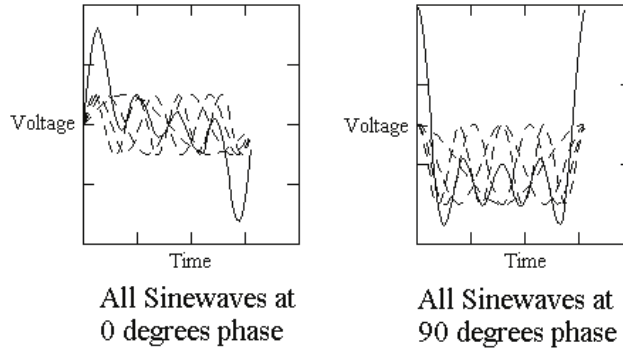


Figure 8-13 Two Multitones with Different Phases

Notice that even though these two waveforms use the exact same components, the 90 degree waveform has a higher maximum peak. This is usually measured by the ratio of the peak voltage to the RMS voltage, which is called the *crest factor*. A waveform with a high crest factor has a high peak voltage relative to the total energy. A waveform with a low crest factor has a low peak voltage relative to the total energy.

Setting all of the waveforms to 90 degrees results in the maximum crest factor, since all the tones are at their peaks simultaneously, right at the beginning of the waveform. Any other combination of phases will have a lower crest factor, but there is no known way to minimize it. Setting all components to 0 phase results in a wave with a fairly low crest factor, although not the absolute minimum.

Usually it is sufficient to leave the phase unspecified and let Makewav2 set them all to 0 degrees. In a few applications, however, using a signal with a high crest factor may be an advantage. Generally, these applications will be where the device-under-test responds to changes in signal amplitude. A high-crest-factor signal will have a short high-amplitude burst followed by a period of lower amplitude, so it

may provide more realistic stimulus than a single tone or a low-crest-factor multitone.

Signal Triggering Considerations

If you are going to use Signal Triggering with your multitone, there are certain restrictions on the multitone. If you violate these restrictions, your multitone may not be recognized by the signal-recognition process. These restrictions also apply to the Frequency Error Correction feature.

The restrictions only apply to a certain range of frequencies, which is the range that is 'searched' by the signal-recognition and frequency correction processes. This range of frequencies varies with sample rate and with waveform length. We define the range as being bounded by two frequencies, F_{high} and F_{low} . Outside this range, there are no restrictions.

Within the range between F_{high} and F_{low} , the following must be true:

- There must be at least five frequency components.
- The frequency components cannot be more closely spaced than six times the generator's frequency resolution value.
- The amplitude of each of these components must be no less than -40 dB from the highest-amplitude tone in the range.

As long as there are at least five frequency components between F_{high} and F_{low} meeting these criteria, the DSP should be able to trigger on your multitone. You can have other tones within the range that are more closely spaced or at lower amplitude, and you may have other tones outside the range, as long as there are five good trigger tones.

The definition of F_{high} is 0.20833 times the sampling rate. The following table shows F_{high} for the most common sampling rates:

Sample Rate	24k	32k	44.1k	48k	54k
F_{high}	5 kHz	6.7 kHz	9.2 kHz	10 kHz	11.2 kHz

Table 10-3 F_{high} for Common Sample Rates

The definition of F_{low} is 0.00625 times the sampling rate. The following table shows F_{low} for the most common sampling rates:

Sample Rate	24k	32k	44.1k	48k	54k
F_{low}	150 Hz	200 Hz	276 Hz	300 Hz	338 Hz

Table 10-4 F_{low} for Common Sample Rates

The following table shows the minimum spacing for the most common sampling rates and waveform lengths:

Waveform Length (samples)	Sample Rate (samples per second)				
	24k	32k	44.1k	48k	54k
8192	18 Hz	24 Hz	33 Hz	36 Hz	40 Hz
4096	36 Hz	47 Hz	65 Hz	71 Hz	80 Hz
2048	71 Hz	94 Hz	130 Hz	141 Hz	159 Hz
1024	141 Hz	188 Hz	259 Hz	282 Hz	317 Hz
512	282 Hz	375 Hz	517 Hz	563 Hz	633 Hz
256	563 Hz	750 Hz	1034 Hz	1125 Hz	1266 Hz

Table 10-5 Minimum Spacing for Common Sample Rates

Creating the ADX file

The starting point to create a custom multitone waveform is entering the desired frequency, amplitude, and (optionally) phase information into the APWIN Data Table. When the information has been created, use the File Export menu command to create an ASCII export (.ADX) file. Alternately, any ASCII text editor could be used if the .ADX file format including the header lines is duplicated.

It is best to begin by setting the parameters on the Sweep panel to the units which you wish to use, as the Data Editor will use these units and will not allow you to change them. Set the Source 1 to one of the frequency settings, such as Gen.Freq. Set the Data 1 to a voltage measurement, such as Anlr.Level A. The units of Data 1 should be volts if you wish to use a linear scale, or any decibel unit (such as dBV, dBu, dBr, etc.) if you wish to work in decibels. Which decibel unit you use is not important since it is used only to specify the relative amplitude of the sinewaves to be generated. The absolute amplitude will be determined when MAKEWAV2 is run.

If you wish to also specify the starting phase of each sinewave to be generated, Data 2 must be selected as the Anlr.Phase, with units in degrees. If you are willing to have all signal components start at zero degrees phase, it is not necessary to select a Data 2 instrument and parameter.

Then bring up the Data Editor and type one row in the Data Table for each sinewave desired in the multitone signal. Use a minus sign for negative decibel or phase values. Values without a minus sign will be interpreted as positive, or you may enter an explicit plus sign. Use the right mouse button in the Data Editor window to obtain a menu of options such as inserting and deleting rows of data.

When complete, use File Export to save the file to disk with a descriptive name and the .ADX file extension. Save it in the directory where APWIN.EXE and MAKEWAV2.EXE reside. If you installed using the default paths, this should be in C:\AP\Apwin150 or C:\Program Files\Audio Precision\Apwin150, depending on whether you are using Windows 3.1 or Windows 95.

The five-frequency file illustrated below was saved as FIVE.ADX.

```
untitled, 09/11/97 12:31:19
Gen.Freq,           Anlr.Level A,           , , , , , ,
Source 1,           Data 1,           , , , , , ,
Hz,                 dBV,              , , , , , ,
70,                 1,                , , , , , ,
270,                1,                , , , , , ,
1020,               1,                , , , , , ,
3920,               1,                , , , , , ,
15000,              1,                , , , , , ,
```

At this point, use MAKEWAV2 as described below from the Windows icon or directly from the DOS prompt.

Running MAKEWAV2

MAKEWAV2 operates with a data (.ADX) file as input and creates two output files. One is a binary waveform file which will be used for generation of the multitone signal. The second is an ASCII .ADF file (meaning data fundamentals) containing a list of the exact fundamental frequencies which make up the multitone signal. This is an intermediate step toward production of the sweep table (.ADS) which will be used during data analysis.

MAKEWAV2 is run by clicking on the icon in the Audio Precision APWIN Program Group and supplying the name of the .ADX file to use as input (plus any desired command line options as discussed below). MAKEWAV2 may also be run from the DOS prompt by typing MAKEWAV2 followed by the .ADX file name to be converted and any command line options.

After all the command line options and the name of the .ADX file have been entered, it is a good idea to redirect MAKEWAV2's output to a text file, so that you can keep a record of the tones present in the file and the waveform statistics, such as the headroom and absolute amplitude. To do this, use the 'greater than' symbol ('>') followed by the name of the text file. It is best to use the same name as the name you used for the .ADX file, except change the extension to '.TXT'.

For example:

```
MAKEWAV2 FIVE > FIVE.TXT.
```

If you chose the Makewav2 icon from the Audio Precision APWIN Program Group, you don't need to type the word 'MAKEWAV2', only the other text that follows it.

It is not necessary to supply the .ADX file extension for the input file nor file names for the output files, if you are willing for the output files to carry the same names as the input .ADX file except for the file extension. In the example above, MAKEWAV2 will automatically name the two output files FIVE.AGM and FIVE.ADF.

The .ADF file for the above example is reproduced below:

```
untitled, 09/11/97 12:31:19
Gen.Freq,           Anlr.Level A,           ,
Source 1,           Data 1,           ,
Hz,                 dBV,           ,
70.3125,            1,           ,
269.531,            1,           ,
1019.53,            1,           ,
3919.92,            1,           ,
15000,              1,           ,
```

Note that the requested frequencies have been rounded off by MAKEWAV2 to the nearest "legal" frequency (integer multiple of 5.8593772 Hz for the default 48 kHz sample rate and 8,192 sample record). Every sinewave must go through an exact integral number of cycles in the generator buffer so that no discontinuity is created when the generator "splices" from the end to the beginning of the record. Control can then be shifted to APWIN and FIVE.AGM can be selected as the generator waveform file.

To create a matching sweep table, use the File Import command to load FIVE.ADF into the data space. The same units used when creating the original .ADX file should be chosen on the Sweep panel as Data 1, Data-2, and Source 1 to avoid an error message when the .ADF file is loaded. The File Save As Sweep Table command can now be used, a file name furnished (which should probably be the same

name as used for the other files, but with the extension 'ADS'), and the Save button clicked to make a sweep table. This sweep table can now be selected from the Sweep panel.

If you wish the output files to have different names, you may supply those in the Parameters field when running MAKEWAV2 under Windows or on the command line when running MAKEWAV2 from DOS. However, file management is normally simplified by using the same file name for the original .ADX file, the waveform (.AGM), fundamental frequency list (.ADF), and the resulting sweep (.ADS) files containing fundamental frequencies.

MAKEWAV2 makes a large number of complex computations in generating the .AGM file. The speed with which it runs depends strongly upon the type of computer used and presence or absence of a math co-processor in the computer. It also depends upon the number of signal frequencies specified, plus several of the options described below. Execution time of MAKEWAV2 can range from a few seconds for .ADX files containing only a few frequencies to a few minutes for complex files. Operation speed can be improved by use of some of the command line options.

MAKEWAV2 Command Line Options

There are a number of options for operation of MAKEWAV2, summarized below. These options may be viewed on screen by double-clicking the MAKEWAV2 icon and then clicking the OK button without supplying anything in the Parameters box, or from DOS by typing MAKEWAV2 /H (for help) or MAKEWAV2 /?.

- /L# Waveform repetition length (lowest period) in samples. (default and max = 8192).
- /R # Sampling rate in Hz. (default = 48kHz).
- /M # Margin (Headroom) to allow, in dB (default = 1.0).
- /S # Record size in samples (file length)
- /A # Absolute amplitude (in dBFS) in output file of a unity (one Volt or zero dB) value in input .ADX file
- /W Cancel waveform output (Do only .ADF file).

`/O #` Oversample ratio. (default = 8).

`/F #` Number of coefficients (Taps) in filter. MUST BE ODD.
(default = 401).

To use one or more of these options, simply type them (separated by one space) on the command line, following the file name to be processed. For example, using FIVE.ADX as the input file and specifying the waveform length, record size, and sample rate values would be accomplished as follows:

```
MAKEWAV2 FIVE /L2048 /S8192 /R32000 > FIVE.TXT.
```

Note that if you select the Makewav2 icon from the Audio Precision APWIN Program Group, you don't need to type the word 'MAKEWAV2', only the options that follow it.

The `/W` (Cancel Waveform Output) option is used to speed up the process if a waveform file is not needed. The ADF file will still be created, and it may be used to determine the exact frequency values that can be generated with the given sample rate and repetition length.

The `/O` and `/F` options are used to modify the algorithm used to compute the peak values of the resultant waveform. They may be modified to speed up execution, or to make more accurate estimation of the peak value of the waveform. Consult your APWIN User's Manual for more detail on these options.

8 Makewave

APPENDIX A - Multitone Program Features

Feature	Options	System One <i>FASTTEST</i>	System One <i>FASTTRIG</i>	System One CODEC	System Two <i>FASTTEST</i>
Measurement Modes	Spectrum	✓	✓	✓	✓
	Response	✓	✓	✓	✓
	Distortion	✓	✓	✓	✓
	Noise	✓	✓	✓	✓
	Masking			✓	✓
	Crosstalk				✓
FFT Length	16384				✓
	8192				✓
	4096	✓			✓
	2048	✓			✓
	1024	✓			✓
	512	✓			✓
	256	✓			✓
	Auto 2xGen		✓	✓	✓
FFT Window	None	✓	✓	✓	✓
	Hann	✓	✓	✓	✓
	Flat-top	✓			
	Blackman-Harris	✓			

APPENDIX A - Multitone Program Features

Feature	Options	System One <i>FASTTEST</i>	System One <i>FASTTRIG</i>	System One CODEC	System Two <i>FASTTEST</i>
Synchronous Processing		✓	✓	✓	✓
Frequency Error Correction			✓	✓	✓
Frequency Resolution		✓	✓	✓	✓
Interchannel Phase		✓	✓	✓	✓
Triggering	Free Run	✓	✓	✓	✓
	Channel 1 Edge	✓			
	Channel 2 Edge	✓			
	Auto Edge	✓			
	DGen	✓	✓	✓	✓
	External		✓	✓	✓
	Signal		✓	✓	✓
Trigger Delay	Selected Times		✓	✓	
	Entry Field				✓

APWIN Simplified for System Two

Book Three



Tutorial

Getting Started with Multitone Testing

Getting Started with Procedures

Introduction

Audio Precision has always believed that automation can be easy. While we have never skimped on programming power and flexibility, we firmly believe that a person should not need a degree in Computer Science to set up an automated series of tests.

APWIN Basic continues this tradition, offering powerful capabilities without the burden of learning a complicated programming language. Learning everything there is to know about APWIN Basic is indeed a large task, as the language has so much to offer. However, we believe that many of you can learn all you need to know in less than an hour, and be ready to set up your own automated tests.

This tutorial is designed to get you started. It will take you through the main features of APWIN Basic and show you how to do the most common tasks without getting bogged down in the advanced features. If you want to continue and learn more about the language, you will have a good foundation.

First, you will learn how to set up a series of automated tests and create a log file of your test results. From there, we will proceed to putting up prompts to give messages to the user. Next we will show how to automatically set the log file options. Then you will learn how to create re-usable building blocks so that you don't have to rewrite the same thing over and over. At this point you will have all the tools you need to set up complete automated test processes.

The next two chapters deal with generating reports (with mixed words, readings, and graphs) and creating menus. These are slightly more complicated topics because the jobs are more difficult. They will take a little more time than the first few chapters, but don't be frightened - you will find the ideas very simple.

Every chapter uses concepts presented in the chapters before it, so it is a good idea to go through them in order.

Getting Started

What Is a Program?

A program is just a list of instructions for a computer to follow. The computer knows how to read these instructions, and by following them, accomplish some sort of task. The instructions are always followed in order, and they must be very specific. They must be written in a special language so the the computer can understand them.

APWIN understands a special-purpose language called APWIN Basic. It is closely related to Visual BASIC, and both are derived from a very old computer language call BASIC, which stands for Beginner's All-purpose Symbolic Instruction Code. The original BASIC was very primitive, but because it was simple to use, several very powerful languages have evolved from it.

Using APWIN Basic, you can easily write a program to perform almost any task that APWIN can do. Almost anything that can be accomplished with the keyboard or mouse can be automated, plus there are many new capabilities that automation gives us.

There are two ways to write a program. The first way is to use Learn Mode. Learn Mode is a feature of APWIN which, when enabled, will follow your keyboard and mouse actions and automatically write a program that duplicates these actions. This is a very powerful feature, as it allows you to write programs without knowing anything about the programming language. Even if you know the entire language, Learn Mode may be a faster and easier way to write programs.

The second way is to type in the program yourself, using a text editor. APWIN has a built-in editor for this purpose. In order to do this, you must know how the language is structured and what words to use to instruct it perform the actions you require.

Learn Mode can't do everything. Some features of the language are only available by typing instructions directly into your program. Many programs use a combination of these two techniques; most of the work is done by Learn Mode and then the finishing details are typed in directly. You will find the language very easy to understand.

Once you have written your program, you can instruct APWIN to read you instructions and follow them. This is known as *running* your program.

There are lots of variations on the word ‘program’, like *macro*, *code module*, *procedure*, etc., which you may see in programming books and in our other manuals. These words have specific meanings to advanced programmers, but they are all just lists of instructions for the computer to follow. For the time being, you don’t need to worry about the differences between them. They’re all just programs.


In our previous software, S1.EXE, we called our program a *procedure*. This worked very well for us until people started using the word *procedure* to mean something else (you will see this in Chapter 5).

Because of our ten-year history of using the word *procedure* to mean a program, this terminology still exists in APWIN software and documentation. In order to keep things as simple as possible, this book will use the word *program* to refer to our program, and the word *procedure* will be used with its new meaning, as in Chapter 5.


Keep in mind when you are using the software and the other manuals that the words *macro* and *procedure* generally refer to your program.

Loading the Samples

All of the sample programs used in this tutorial are provided with APWIN. In each diagram, the name of the sample is shown in the title bar of the Procedure Editor. Unless you chose not to install the samples when you ran setup, they should be installed on your hard drive. If you installed into the default directories, you should find them in the directory C:\Apwin\S1\Tutorial\APBasic or C:\Apwin\S2\Tutorial\APBasic, depending on which System you are using. If you changed the installation directories, the exact path may be different.

To load and view a program, select Procedure Show Panel from the menu or press the toolbar icon that looks like this: 

A new window called the Procedure Editor should pop up. It acts a lot like an independent program, and has its own menus, but it is very closely tied to APWIN.

To load a program into the Procedure Editor, select File Open from the menu or press the icon that looks like this: 

A file-choosing window should pop up, which will allow you to choose a program to load. Locate the sample that you wish to load, making sure that you are using the correct sample for the type of System you are using.

The only difference between the programs in the System One and System Two directories is that the extensions to the tests names are different - System One tests use the extension at1, while System Two tests use the extension at2. APWIN runs in two completely different modes, depending on which type of System you use or the choice you make when starting APWIN. It is recommended that you only load tests that were saved with the same type of System you are using. So, choose the directory that matches the current APWIN mode..

Some of our printed examples use System One tests, others use System Two tests. Both kinds are available in the samples directories. If you type them in, however, you need to be aware of which kind to use and adjust your programs accordingly.

It would be a good idea to copy the examples to a different directory - then you can freely change the copies without ruining the originals. If you change the files in the sample directories, and then decide you want the original programs back, you will have to run APWIN Setup again!

Using Learn Mode to Automate Tests

Let's begin by using Learn Mode to automate some tests for us. To do this, you will need a computer running APWIN and a System One or System Two. If you don't have a System, you can still go through the exercise using APWIN's Demo Mode, but the data will be random and rather erratic.

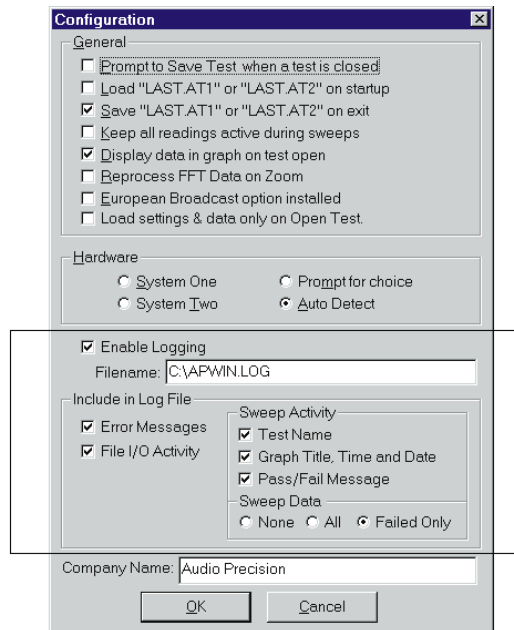
No external hardware or cables are required. If you currently have a device-under-test connected to your System, disconnect it at this time.

We will be using tests provided with APWIN. These tests are not installed if you chose not to install the samples, so you may need to rerun APWIN Setup and have the samples installed.

The first thing we should do is set up the log file options. This will allow our automated series of tests to run unattended, and give us a pass/fail report at the end.

If you have not yet started APWIN, do so now. Then select Utilities Configuration from the main menu. You should see the Configuration panel. At the top of the panel are a number of settings that are not relevant to this exercise - the log file settings are in the bottom half of the panel.

Use the following illustration as a guide, and set all the options within the box the same as shown.



Configuration Panel, Set Up for Test Automation

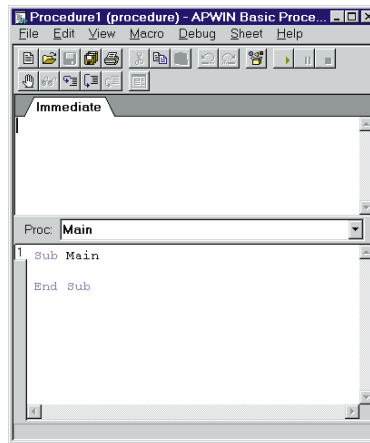
Press OK to accept your settings.

These configuration settings will stay the same the next time you run APWIN, and they are not saved with the test, so they should not need to be set again unless another program changes them. Later we will learn how to use APWIN Basic to make these settings automatically.

Check the Utilities menu. If the Clear Log File menu option is available (not greyed out), select it to clear the log file.

Now select Learn Mode from the Utilities menu. The Procedure Editor window will pop up in front of APWIN. If you move your mouse over APWIN your mouse pointer should change so that it has a small cassette tape on it, to remind you that your actions are being recorded.

The Procedure window should look like this:



Default Procedure Editor

The upper part of this window (marked 'Immediate') can be ignored for now. The bottom part of the window contains the program. You could type into this area to create your program, but we will let APWIN write the first program for us.

There are already some instructions in the bottom part of the window, the words 'Sub Main' and 'End Sub'. These instructions define the

beginning and end of your program. Any instructions that your program wants to run need to be placed between these two.

Now go back to APWIN by minimizing the Procedure Editor or by clicking on the APWIN title bar. From the menu, select File Set Working Directory. A file browser should pop up, allowing you to select a directory (or folder) in which the tests can be found.


Find your way to the tutorials directory for the type of System you are using. If you installed to the default directories when you ran APWIN Setup, the files will be in one of the following directories:


System One: **C:\Apwin\S1\Tutorial\APBasic**

System Two: **C:\Apwin\S2\Tutorial\APBasic**

If you have located the correct directory, you should see the files Phase.at1 (for System One) or Phase.at2 (for System Two) and Response.at1 (for System One) or Response.at2 (for System Two).


Once you have located the directory containing these files, press OK.

Next, Select File Open Test or press this button on the main APWIN toolbar: 

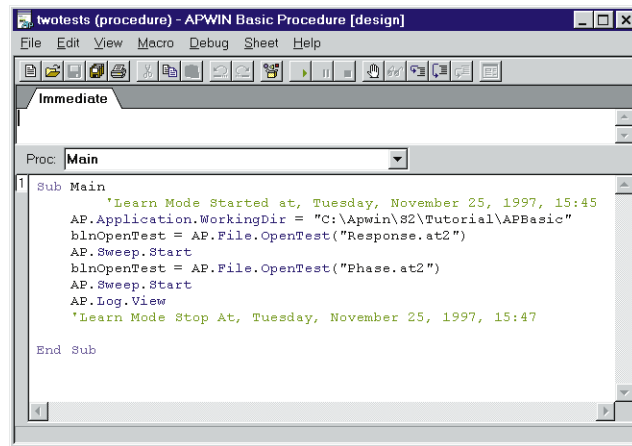
The file browser should pop up, showing the contents of the directory chosen above. Choose the response test, which will be called Response.at1 or Response.at2. Then start the sweep by selecting Sweep Start from the main menu, pressing , or pressing the Go button on the Sweep panel or main toolbar. Wait for the sweep to finish.

Next, load the phase test. This will be in the same directory as the Response test loaded earlier, and will be called Phase.at1 or Phase.at2, depending on which type of System you are using. Start the sweep again and wait for it to finish.

Now select Utilities View Log File. The log file should open, showing the results of the response and phase tests. Close the log file when you are done looking at it, then go back to APWIN and select Utilities Learn Mode from the menu. This should stop Learn Mode and your mouse pointer should change back to the normal arrow.



Bring the Procedure Editor to the front by selecting Panels Procedure Editor from the menu or by pressing this toolbar button: .

It should look like this:



Program to Run Two Tests

Between the 'Sub Main' and 'End Sub' are now the instructions to load and run these two tests and view the log file. If you are using System One, the test names will be different.

Run the program by pressing , selecting Macro Run from the menu, or by pressing this toolbar button: .

The two tests should run again and the log file should pop up. The log file should now show two runs of each test.

Take note of what the editor panel looks like when your program is running. Notice that the 'Run' button goes gray, and then turns green again when your program is finished. This is how you can tell whether your program is running. You will find this very useful when you are writing and testing your programs.

If you select View Procedure Toolbar from the APWIN menu, a copy of this toolbar will be shown on the main APWIN screen. Either toolbar may be used interchangeably.

What If I Make a Mistake?

If you hit the wrong key or menu item while Learn Mode was activated, APWIN will dutifully record and recreate your error. In the next chapter you will learn about editing the program, and you may be able to use the editor to erase or fix the problem. However, it may be easier to start over and relearn the program.

To start over with a clean program, go to the Program Editor window and select **File** **C**lose from the menu. A window will pop up asking if you want to save the changes to your program. Press 'No', and your existing program will be replaced with an empty one. You may then restart Learn Mode and learn the program again.

Adding to A Program

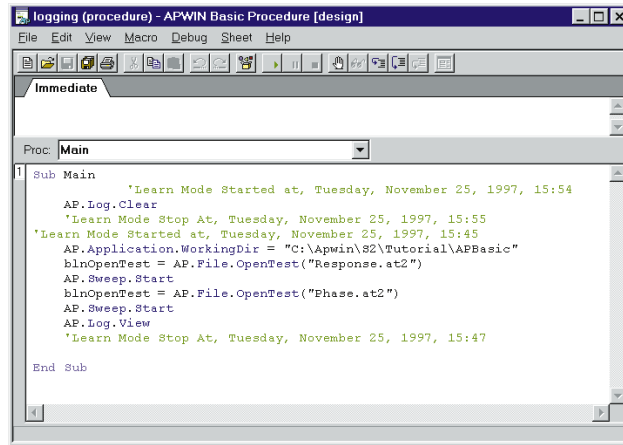
Learn Mode can also be used to add instructions to an existing program.

It is usually more convenient to clear the log file before each pass through the test suite, so that the only results shown in the log file are from the latest pass. Let's use Learn Mode to add this feature to our program.


Go back to the Procedure Editor. If you click your mouse anywhere within the editor pane (where the instructions are) you should see a flashing vertical bar called the *cursor*. If you type into the editor, the cursor is where the typing will be inserted. If you invoke Learn Mode, the new instructions learned by APWIN will be placed at the cursor position.

We want the new instruction to be placed at the beginning of the program, so position the cursor at the beginning of the line that says "AP.File.OpenTest". Then go back to APWIN and start Learn Mode by selecting **Utilities** **L**earn Mode from the menu.

Select Utilities Clear Log File from the menu. Then stop Learn Mode by selecting Utilities Learn Mode from the menu. Go back to the Procedure Editor, which should look like this:



Program to Clear Log and Run Two Tests


Run the program by pressing **F5**, selecting Macro Run from the Procedure Editor's menu, or by pressing this toolbar button: 


The two tests should run again and the log file should pop up. The log file should only show the results from the latest pass through the test suite.

At this point, you may want to save your work. To do this, select File Save or File Save As from the Procedure Editor menus or press the floppy disk button on the Procedure Editor toolbar. You will need to select a name for the program file. I called mine 'logging'.

You can also save your program from the main APWIN menus by selecting File Save Procedure or File Save As Procedure. Note that the floppy disk button on the main APWIN menu will not save your program, only the current test.

Once you have saved your program, you can reload it in several ways. One way is to use the main APWIN menus to select File Open Procedure, and then selecting the program name. The program will

load invisibly, but you can run it by selecting Procedure Run from the APWIN menu or pressing this toolbar button: 

You can view or edit your program by selecting Procedure Show Panel from the APWIN menu or by pressing this toolbar button: 

Using the Editor

In the previous chapter, we used Learn Mode to write a program to load and run several tests. In this chapter, we will use the Procedure Editor to add some features that are not possible using Learn Mode alone. We will start with the program developed in the last chapter.

Before we start editing our program, we will take a few minutes to get to know the Procedure Editor and APWIN Basic.

If You are New to Text Editing

If you are already comfortable with a Windows-compatible text editor or word processor, you will probably find the Procedure Editor very similar, and will have no trouble operating it. You may want to skip the rest of this section.

Look for the flashing vertical bar. This bar is called the *cursor*. If you type, the cursor is where the letters will show up. You can move the cursor with the arrow keys on the keyboard, or by placing your mouse pointer somewhere else within the editor window and pressing the left mouse button.

Usually when you type into the editor, the letters you type are inserted at the cursor position. If you hit the Insert key, you will go into Overtyping mode, where the letter after the cursor is replaced with the letter you typed. Hitting the Insert key again will put you back into Insert mode, which is usually a better mode to use.

The most common way to delete letters is with the Backspace key, which will delete the letter to the left of the cursor. You can also use the Delete key, which will delete the letter to the right of the cursor.

The Home key will move the cursor to the beginning of the line you are on, and the End key will move the cursor to the end of the line.

The Tab key is usually used at the beginning of a line to indent the line. This is usually done to show the structure of the program. To unindent, go to the beginning of the line and press Delete, or go to the beginning of the indented text and press Backspace.

You can delete or copy large sections of text using the Cu~~t~~, Co~~p~~y, and Paste commands on the Edit menu. All of these commands use the *copy buffer*, which is a memory area where a section of text is stored. If you place something in the copy buffer, it will remain there until you store something else in the copy buffer, which will cause the original text to be lost.

To place something in the copy buffer, must *highlight* it. You can highlight text by dragging your mouse over it while holding down the left mouse button, or by holding down the shift key while using the arrow keys. The highlighted text should appear on a black background instead of a white background.

After you have highlighted some text, you can use the Edit Co~~p~~y or Edit Cu~~t~~ commands to place it in the buffer. Edit Co~~p~~y will place a copy of the highlighted text in the buffer without changing the text in the editor. Edit Cu~~t~~ will remove the highlighted text and place it in the buffer.

Once you have used Edit Cu~~t~~ or Edit Co~~p~~y to fill the copy buffer, you can use Edit Paste to insert a copy of the buffer in the editor at the cursor position.

For example, if you wanted to duplicate a line a number of times, highlight it and then select Edit Co~~p~~y. Then place the cursor on the next line and select Edit Paste. Repeat the Paste as many times as you like. You do not need to repeat the Co~~p~~y.

If you have highlighted text, and then you type, the letters that you type will replace the highlighted text, so the highlighted text will be erased.

You can also indent a section of text by highlighting it and then pressing the Tab key. You can un-indent a section of text by highlighting it, holding down the Shift key, and pressing the Tab key.

One thing that is always confusing when learning about text editors is how to create a new line to type on. To do this, go to the beginning or end of a line and press Enter (also called Return). To erase a blank line, go to the beginning of the line after it and press backspace. To join two lines together, go to the beginning of the second line and press backspace.

You may want to move or resize your editor window. To move it, place the mouse pointer on the title bar (the blue strip at the top of the window), hold down the left button, and move the mouse to move the window. Release the button when the window is correctly positioned.

To resize the window, move the mouse pointer down to the bottom right corner of the window and find the spot where it changes to a diagonal arrow. Then hold down the left mouse button and move the mouse to change the size of the window. Release the button when it is the size you want it.

The Numbers on the side of the Editor Window

You may have noticed some numbers (or perhaps only one) along the left side of the editor window. These numbers are sort of like page numbers. You can have a number of programs in your editor at any time, and you can click on these numbers to change pages. To get a new page, click the 'New Program' button.

Any time you Run or Save, you will only Run or Save the current page. The current page is the one that you are currently looking at, which appears to be on the front of your book.

This is a very useful feature if you are working on a program and you want to take a look at a sample that does something similar - just open another page (using the New Program button), load the sample, look at what you want, and then come back to your program page.

You can only open nine different programs at the same time. If you try to open a tenth program, you will get the error message "Can't Open Another Sheet.". You will have to close one of the programs that is already open before you can open another one. To close a program, select File Close from the menu.

Play, Pause, and Stop Buttons

As you probably noticed, there are also Stop and Pause buttons near the Run button. The Stop button has a red square on it, and the Pause button looks like two vertical bars. They function just like most cassette player (or more recently, CD player) buttons. Pause will halt your program temporarily, but if you press Run again your program

will pick up where it left off. The Stop button will stop your program entirely, and you will have to start over at the beginning.

Normally if you encounter an error, APWIN will pause your program. It won't let you edit your program while it is paused, so you will have to hit Stop before you can fix your program.

Reading Programs

In order to make any changes to a program that has been created using Learn Mode, you need to be able to read the program, so that you can tell how it is operating and what changes are necessary.

APWIN Basic is fairly easy to read. Like any computer program, an APWIN Basic program is just a list of command for the computer to follow. Each command performs some small part of the larger task to be accomplished. When you run the program, the computer will read the instructions in order, and follow each one. There is always one command per line.

Most commands begin with the letters 'AP'. These commands tell APWIN to perform some action, just as if you had selected a menu item or pressed a toolbar button. The commands that don't start with 'AP' are not direct instructions for APWIN, but rather accomplish some task for the program itself. Examples of this are the 'Sub Main' and 'End Sub' commands at the beginning and end of the program. These commands tell the program where to start reading commands and where to stop.

After the letters 'AP' there is usually a period, and then another word which defines the category of the command. Then there is usually another period and the exact command within that category. The AP, the category, and the exact command are all part of the command word.

If more information is needed to complete the instruction, it is enclosed in parentheses or quotes (or both) at the end. If there are several pieces of data needed, they will be separated by commas and the entire group will be surrounded by parentheses.

Most of the commands are probably obvious, but if you see a command and you don't know what it does, you can look it up in the

APWIN Help. The AP commands can be found in the Procedure Editor menus under Help APWIN Extensions Help or from the APWIN menus under Help APWIN Basic Extensions. The non-AP commands can be found in the Procedure Editor menus under Help Language Help or from the APWIN menus under Help APWIN Basic Language. You will probably want to use the Search feature to locate the command alphabetically.

You can also use the Help system to find a command to perform a certain task, but it may be easier to invoke Learn Mode, perform the action, and look at the Procedure Editor to see what Learn Mode wrote.

Once you have used Learn Mode to write a program, it should be fairly easy to look at the Procedure Editor and determine where changes need to be made.

Let's take a look at the last program we created in the last chapter and see what it is doing. The program should look like this (remember that you can load the program 'logging' from the sample directory):

```

logging (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help
-----
Immediate
Proc: Main
Sub Main
    'Learn Mode Started at, Tuesday, November 25, 1997, 15:54
    AP.Log.Clear
    'Learn Mode Stop At, Tuesday, November 25, 1997, 15:55
    'Learn Mode Started at, Tuesday, November 25, 1997, 15:45
    AP.Application.WorkingDir = "C:\Apwin\82\tutorial\APBasic"
    blnOpenTest = AP.File.OpenTest("Response.at2")
    AP.Sweep.Start
    blnOpenTest = AP.File.OpenTest("Phase.at2")
    AP.Sweep.Start
    AP.Log.View
    'Learn Mode Stop At, Tuesday, November 25, 1997, 15:47
End Sub
  
```

Program to Clear the Log File and Run Two Tests

The first line of the program says 'Sub Main'. This is required to show the place where the program begins.

The next line says 'Learn Mode Started at...'. Usually this line will be highlighted in green. This line doesn't cause APWIN to perform any action. The APWIN Basic interpreter will ignore it. In fact, you could remove this line entirely and your program would still run perfectly. It is what is called a *comment* - a note inserted into your program to help the reader understand what the program is doing. A comment always begins with a single quote character ('). Any line that begins with a single quote character will be ignored. This particular comment line is inserted by Learn Mode to show where the learning session began.

The following line says 'AP.Log.Clear'. This line tells APWIN to clear the log file, just as if you had selected Utilities Clear Log File from the menu.

The next two lines are also comments. The first one indicates the end of a learning session, and the second indicates the beginning of another session.

The next line starts with `'AP.Application.WorkingDir = '`, and tells APWIN to change the working directory.

After that is a line that performs the Open Test operation. It begins with the word `'bInOpenTest'`. The first part of this line, up to the equals sign (`=`), doesn't perform any useful function in this program, so it can be ignored or even erased. The part after the equals sign actually performs the action of opening a test file.

The next line starts the sweep. The program does not continue until the sweep is finished.

The next two lines perform a test load and run the sweep for the next test.

Following those is the line `'AP.Log.View'`, which will view the log file.

Then there is a final comment indicating the end of the learning session, and the line `'End Sub'`, which indicates the end of the program.

When is a program just like another?

While the language interpreter is very picky about some things, there are other things which make not difference whatsoever. For example, blank lines are ignored. If you see blank line in a program, you can put in no blank lines or a hundred blank lines. The language interpreter is just going to skip over them anyway.

Capitalization is important to APWIN, but it will often fix your capitalization for you. There are times when it can't fix it, so it is best to keep an eye on your capitalization to reduce the risk of errors.

Indenting is also ignored by the language interpreter. Most experienced programmers use the tab key to indent parts of their programs to point out how the program is structured. Since your programs aren't really structured yet, I would suggest you just follow my indenting until it becomes obvious what is going on. APWIN won't care, though, so if you want to just forget indenting and start each line at the very left, or leave it wherever Learn Mode puts it, you can do that.

A Trick: If you want to indent a number of lines of your program, first highlight them by dragging your mouse over them (they should

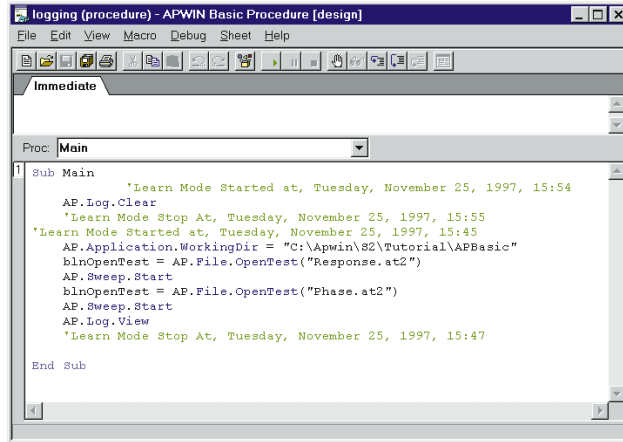
turn black) and then hit the Tab key. To un-indent a number of lines, highlight them and then hold down the Shift key and hit Tab.

Almost everything else in your program needs to be identical to the examples. Every word must be spelled exactly correctly and must be separated from other words by spaces or a comma. Many words have periods in them, so be careful.

You can almost always change things that are between double quote marks (“), such as the file name in an ‘OpenTest’ command. In fact, in many cases you will have to change the filename extension or the test file name in order to make the program work for your situation.

Aligning Your Program

The first thing you may want to do after using Learn Mode is to align the commands in your program, as Learn Mode tends to create unusual indentation. For an example, let's look at the program we created in the previous chapter. It should look like this:



```
logging (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help
Immediate
Proc: Main
Sub Main
    'Learn Mode Started at, Tuesday, November 25, 1997, 15:54
    AP.Log.Clear
    'Learn Mode Stop At, Tuesday, November 25, 1997, 15:55
    'Learn Mode Started at, Tuesday, November 25, 1997, 15:45
    AP.Application.WorkingDir = "C:\apwin\82\tutorial\APPBasic"
    blnOpenTest = AP.File.OpenTest("Response.at2")
    AP.Sweep.Start
    blnOpenTest = AP.File.OpenTest("Phase.at2")
    AP.Sweep.Start
    AP.Log.View
    'Learn Mode Stop At, Tuesday, November 25, 1997, 15:47
End Sub
```

Program to Clear the Log File and Run Two Tests

Notice that one line is overindented and another is underindented. To fix the overindented line, place the cursor at the left edge of the editor pane, on that line, and then press Delete. To fix the underindented line, place the cursor at the beginning of the line and press Tab.

The indenting does not matter to APWIN, but makes the program structure much more apparent to the person reading it. For example, the indenting we have done makes it clear that the instructions are enclosed by the 'Sub Main' and 'End Sub' commands, and that the entire program is one logical block. This will become more important as we develop more complex programs.

Making A Program Portable

All of the programs we have created so far used *absolute paths*. This means that they refer to a specific directory on your hard drive. If the tests that you want to run are not in the specified directory, your program will not run.

What if you decide to move your program and tests to a different directory? Or give your program to someone else, who may not want to put it in the same directory as you did? If you do either of these things, the program will not work as it is.

The absolute path is specified in the 'AP.Application.WorkingDir = ' command. This tells APWIN where to find your test files. It specifies the exact drive, and the exact path on that drive, where the tests are located.

The easiest way to make your program 'portable', so that it can be moved around from directory to directory, is to place all the test files in the same directory as the program. You can then tell the 'AP.Application.WorkingDir' command to look for the test files in the same directory as the program file. To do this, change the AP.Application.WorkingDir command to look like this:

```
AP.Application.WorkingDir = MacroDir
```

The word 'MacroDir' refers to the directory from which the program was loaded. APWIN will then expect to find the test files in that directory. Note that there should be no quotes around the word 'MacroDir' in your program.

If you use this command instead of specifying the absolute path, you can locate the program and all of its associated files in any directory, and even move it around from directory to directory, as long as the program and all the test files (and any other necessary files, such as limit or waveform files) are moved together, and located in the same directory. The rest of the examples in this tutorial use this method to set the working directory.

Creating Prompts

The next thing we will learn is how to put up a prompt. A prompt is a message to the person running the program. The program will stop while the person reads the message, and won't continue until a key or mouse button is pressed.

Most often, a prompt will be used in a test suite to tell the operator to change cabling or make changes to the device-under-test.

Creating a prompt requires three commands. The first command looks like this:

```
AP.Prompt.Text = "Change Cables Now."
```

The message between the quotes can be replaced with any message that you wish to display. This command declares the text that will be displayed for the prompt.

```
AP.Prompt.ShowWithContinue
```

This command tells APWIN to show the prompt with the previously declared text. The prompt window will have a Continue button, to allow the operator to continue with the procedure after reading the message.

```
Stop
```

This command tells APWIN to pause and wait for the signal to continue. The continue signal will be automatically generated when the operator presses the Continue button

Let's add a prompt between the two tests in the program we have been working on. If you are not already there, go to the Procedure Editor and load the program.

Then position the cursor at the beginning of the line that loads the phase test. Hit the Enter (or Return) key five times to create five blank lines in your program. Type the three lines shown above into the blank lines, leaving a blank line before and after the three new lines.

Your program should now look like this:

The screenshot shows a window titled "prompt (procedure) - APWIN Basic Procedure [design]". The menu bar includes File, Edit, View, Macro, Debug, Sheet, and Help. Below the menu is a toolbar with various icons. The main area is divided into an "Immediate" window at the top and a code editor below. The code editor shows a procedure named "Main" with the following code:

```

1 Sub Main
  'Learn Mode Started at, Tuesday, November 25, 1997, 15:54
  AP.Log.Clear
  'Learn Mode Stop At, Tuesday, November 25, 1997, 15:55
  'Learn Mode Started at, Tuesday, November 25, 1997, 15:45
  AP.Application.WorkingDir = MacroDir
  blnOpenTest = AP.File.OpenTest("Response.at2")
  AP.Sweep.Start

  AP.Prompt.Text = "Change Cables Now."
  AP.Prompt.ShowWithContinue
  Stop

  blnOpenTest = AP.File.OpenTest("Phase.at2")
  AP.Sweep.Start
  AP.Log.View
  'Learn Mode Stop At, Tuesday, November 25, 1997, 15:47

End Sub

```

Program to Run Two Tests with Prompt for Cable Change

Run the program. The response test should run, and then a window should pop up informing you to change the cables. There aren't actually any cables to change - this is just an example. When you press Enter (or Return) or use the mouse to press the Continue button, the phase test should run. Then the log file should appear, showing the results of the two tests.

Log File Options

If you have followed the tutorial this far, you have created your own program and made it run some tests for you, reporting the results to a log file.

Because the log file options cannot be changed using Learn Mode, we had to set them using the Utilities Configuration panel. If we took our program to another computer, and the log options were not set correctly, we may get incorrect log file output or no log file at all.

In this chapter we will see how to have the program set the log file options, so that every time we run the program the log file output will be the same, no matter what computer we run it on.

In the process, we will learn about properties.

What is a Property?

A property is some attribute of APWIN that can be changed by the operator or by a program. For example, the analog generator amplitude is a property. Either the user or a program can put a new value for the analog generator amplitude, and APWIN will use that value until a new value is assigned to that property.

Every APWIN property that is accessible to your program has a name. The names are similar to the command names, consisting of the letters 'AP', followed by a period and the category of the property, then another period and the name of the property.

Properties are assigned using the equals sign. On the left of the equals sign is the property, and on the right of the equals sign is the value to assign it to. The value can be many different things, depending on the type of property. Some properties require numerical values, while others require words or True/False values.

For example, the property `AP.Gen.Output` determines whether the analog generator is on or off. Our program can turn the generator on by issuing the following command:

```
AP.Gen.Output = True
```

Many properties that require numerical values also require the units to be specified. The requested units must be placed in parentheses and quote marks just to the right of the property name, but left of the equals sign. For example, we could use the following command to set the analog generator amplitude to 2 Volts RMS:

```
AP.Gen.Ampl( "Vrms" ) = 2
```

Any value that is not True/False or numerical must be enclosed in quote marks.

Log File Properties

Each of the options in the Utilities Configuration panel has some property associated with it, and your program can control all of them.

Any check box will be a True/False property. Setting the property's value to 'True' will be the same as placing a check mark in the box, and setting the value to 'False' will be the same as removing the check mark from the box. The following commands work this way:

```
AP.Log.ErrorMessages  
AP.Log.TestName  
AP.Log.FileActivity  
AP.Log.GraphTitle  
AP.Log.PassFailMessages
```

For example, if you don't want your log to show every test being loaded from disk, you could put this command in your program:

```
AP.Log.FileActivity = False
```

The multiple-choice buttons that choose what data you want sent to the log file are a property called `AP.Log.Data`. This property requires a numerical value of 0, 1, or 2. The choices go in order - 0 is None, 1 is All, and 2 is Failed Only. So, if you wanted your program to set it to log none of the sweep data, you would put the following line in your program:

```
AP.Log.Data = 0
```

The property `AP.Log.FileName` is used to set the name of the file to which the log information will be sent. Using this option, you can have the information from different parts of the program go to different log files, or have it all sent to a special file. The value assigned to this property is a filename and path, not a True/False or number, so it must be enclosed in quotes, like this:

```
AP.Log.FileName = "C:\Temp\Logfiles\APWIN.LOG"
```

You must be very careful when you do this, because if you specify a directory that doesn't exist, you will get an error message. This means that your program might run fine on your computer but not on someone else's, because the directory exists on your computer but not on theirs. The only directory guaranteed to exist is "C:\", so if your program is going to run on any computer, it should store its log files there.

Another thing you can do with your log files is to add text to them. You can use this to add whatever comments you want. To do this, use the following command:

```
AP.Log.AddEntry "My Program Works Beautifully"
```

This message will be written into the log file as soon as APWIN Basic encounters this instruction.

Remember: watch the order of your instructions

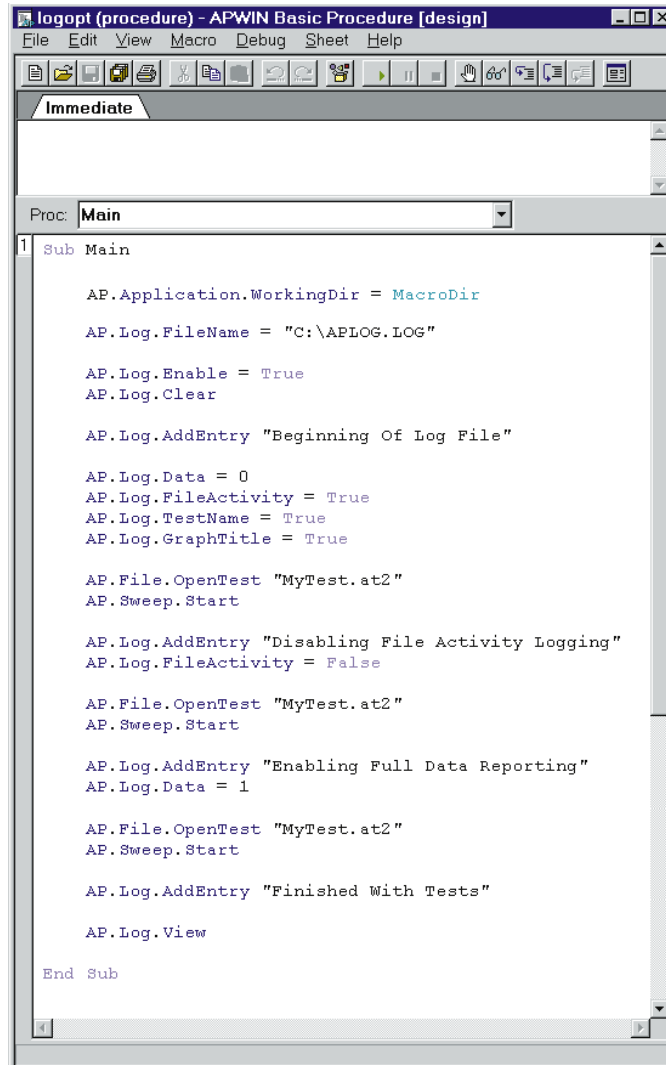
Don't forget that APWIN is going to read the file in order, from top to bottom, and do each thing as it goes along. If you run some tests, and then enable the log file, your tests will not be in the log. Similarly, if you change any options after running some tests, those options will not apply to the tests run previously. Usually it is best to set all the options

you want at the very beginning, including enabling the log file, then run your tests, and then view the log file at the very end.

However, if you want to run a test but you don't want it to appear in the log file, you can disable the log file before running the test and then enable it afterward. Similarly, if most of your tests use limits, but one test does not, you may want to set APWIN to report all the data to the log file before running this test, and then set it back to one of the other options afterward.

Example of Log Options

The following is an example that uses some of the commands and properties described in this chapter. If you're not sure what the program is doing, look at the AP.Log.AddEntry commands in the program. I am using these commands, which add comments directly to the log file, to explain the configuration changes I am making.



The screenshot shows a window titled "logopt (procedure) - APWIN Basic Procedure [design]". The window has a menu bar with "File", "Edit", "View", "Macro", "Debug", "Sheet", and "Help". Below the menu bar is a toolbar with various icons. The main area of the window is divided into two sections: "Immediate" at the top and a code editor below. The code editor shows the following code:

```
1 Sub Main  
  
    AP.Application.WorkingDir = MacroDir  
  
    AP.Log.FileName = "C:\APLOG.LOG"  
  
    AP.Log.Enable = True  
    AP.Log.Clear  
  
    AP.Log.AddEntry "Beginning Of Log File"  
  
    AP.Log.Data = 0  
    AP.Log.FileActivity = True  
    AP.Log.TestName = True  
    AP.Log.GraphTitle = True  
  
    AP.File.OpenTest "MyTest.at2"  
    AP.Sweep.Start  
  
    AP.Log.AddEntry "Disabling File Activity Logging"  
    AP.Log.FileActivity = False  
  
    AP.File.OpenTest "MyTest.at2"  
    AP.Sweep.Start  
  
    AP.Log.AddEntry "Enabling Full Data Reporting"  
    AP.Log.Data = 1  
  
    AP.File.OpenTest "MyTest.at2"  
    AP.Sweep.Start  
  
    AP.Log.AddEntry "Finished With Tests"  
  
    AP.Log.View  
  
End Sub
```

LOGOPT.APB Program

Re-usable Program Pieces (Procedures)

In this chapter we introduce the *procedure*, a very powerful programming tool. The basic idea is that you can take a piece of a program, which performs a certain task, and give it a name. Your program can then use this name to perform the task, instead of repeating the instructions.

Here's an example: Let's say you want to write several dozen programs to test a number of different instruments. Each of them will be reporting to a log file, using the same reporting format. If you didn't know about procedures, you would have to write the instructions for setting up the log file several dozen times.

If you knew about procedures, you might write a procedure which contained all the instructions for setting up the log file, and name it `SetUpLog`. Then each of your programs could just use `SetUpLog` (as though it was another command in the language) to set up the log file. It's like being able to add your own commands to APWIN Basic.

Not only can this save you a lot of typing, but it can make changes much easier. If you decided to change your log file format, you would only have to change this one procedure, instead of going through several dozen programs and changing them all.

You can use procedures for a lot of things besides setting up log files. Any time you have to perform the same task more than once, you should consider making a procedure.

Procedures, like programs, can take on a lot of different names. They can be called sub-procedures, subroutines, functions, and a few other names. These names have slightly different meanings to experienced programmers, but you don't have to worry about the differences between them. For now, you can think of them all as procedures.

Those of you who used procedures in S1.EXE will notice that we have changed the definition of the word. In S1.EXE, we called our programs procedures. Then people started calling these other things procedures, and eventually we were forced to change our language.

Every time a program runs a procedure, it is known as *calling* the procedure.

Here is an example of a procedure:

```
Sub MyProcedure
  AP.Prompt.Text = "My Procedure Works!"
  AP.Prompt.ShowWithContinue
  Stop
End Sub
```

This procedure has four main ingredients:

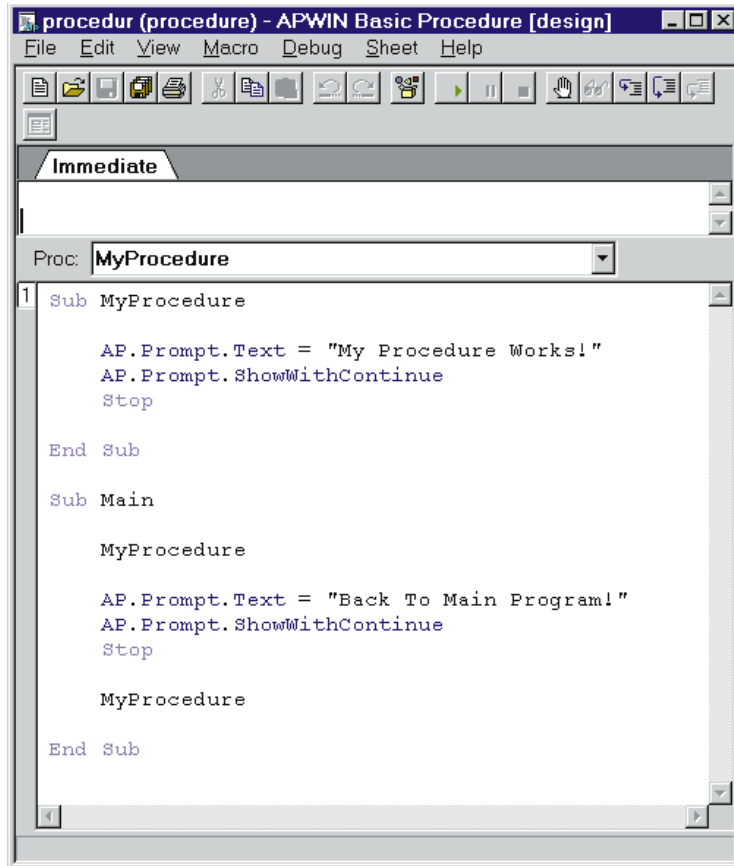
1. A place to start - the word "Sub" indicates that this is the beginning of a procedure.
2. A name for the procedure - in this case, "MyProcedure"
3. Instructions to run when the procedure is called
4. A place to stop - the words "End Sub" indicate that the procedure ends here

Now we can use the word `MyProcedure` as a new command - a command that puts a prompt up on the screen with the words "My Procedure Works!"

Those of you with keen eyes may have noticed something familiar - this procedure I just showed you has the same basic structure as a program. It starts with a `Sub something`, ends with `End Sub`, and has instructions in between. In fact, a program is just a procedure called `Main`. When you push the go button, the language interpreter will scan the program for a procedure called `Main` and run it. In fact, even if there are other procedures, `Main` is the only one that will be run unless `Main` calls one of the other ones. In this way, `Main` is in charge of what gets run. If this is confusing, don't worry about it, you can continue on without understanding this paragraph.

A Simple Procedure

I think we're ready for an example of a program using a procedure. This first example shows the procedure in the same file as the main part of your program - later we will show you how the procedure could come from a different file.



```
1 Sub MyProcedure
    AP.Prompt.Text = "My Procedure Works!"
    AP.Prompt.ShowWithContinue
    Stop
End Sub
Sub Main
    MyProcedure
    AP.Prompt.Text = "Back To Main Program!"
    AP.Prompt.ShowWithContinue
    Stop
    MyProcedure
End Sub
```

PROCEDUR.APB Program

Try this program. Be sure to save it. If you have typed it in correctly, it should give you the message "My Procedure Works!", wait for you to hit continue, give you the message "Back to the Main Program!", wait wait for you to hit continue, give you the message "My Procedure Works!" again, wait for you to hit continue again, and then end.

How did we do that?

When you hit the run button (or select it from the menu), the language interpreter scans through the program for a procedure called Main (looking for the words “Sub Main”) and begins reading there. It will follow every instruction it sees, and the first instruction it finds (after “Sub Main”) is the instruction “MyProcedure”. Since it doesn’t recognize this word at first, it will look through the program for a procedure with this name. When it finds the words “Sub MyProcedure”, it knows that this is the procedure that you meant, so starts from there. It will follow the three instructions inside the procedure, which will give you the prompt and wait for you to hit continue.

The next thing it will find will be “End Sub”, so it knows it has reached the end of the procedure. It remembers where in the Main Procedure it left off, so it goes back to there and continues. Next, it goes through the next three lines which put up the “Back to the Main Program!” prompt. Then it sees MyProcedure again, and calls the procedure again, the same as it did the first time. Then it comes back to where it left off in Main, reads the final “End Sub”, and ends.

When you write your own procedures, be careful that the name you choose for your procedure is not a word that is already used by APWIN Basic. If you defined a procedure called Stop, and then used the word Stop, how does it know which one you mean? The new Stop you’ve just defined, or the old Stop that is built-in? Usually, APWIN Basic will spot the conflict and give you an error message, although it won’t usually tell you exactly what wrong (as usual, it can’t really tell). However, if you suspect that the name you used for your procedure is already used by the language, add an X or something to the end of it, because even if Stop is defined, StopX isn’t, so this should get rid of the conflict. Of course, everything that calls that procedure will have to be changed as well.

You can also call procedures from other procedures. Be careful, though, if you have two procedures that both call each other your program could get locked in a circle and never get out!

Building a Library of Procedures

Another thing you can do is put your procedures in a different file from your program. This is easy to do, but you do have to tell your program where to look for the procedures it needs.

This way, you can build a 'library' of procedures in a file, and tell each of your programs about it, and then all of your programs can use the procedures in that file. A single file can have as many procedures in it as you like, and it can be used by as many different programs as you like. A single program can also use any number of library files, but you must tell the program about each one.

The command is as follows:

```
`#uses "ProcLib.apb"
```

The single-quote mark and the number-sign at the beginning are part of the word, so they must be included, and there can be no spaces between them.

When the language interpreter finds this instruction, it knows that there are procedures in this other file that this program may be calling. This way, if you call any procedures it can't find, it knows where to look for them. Hopefully they will be found in this other file. Otherwise, you will get an error message.

You will notice that we didn't give the full path to the procedure file. Since we didn't, it will look in the directory where it found the program, and maybe some other directories (but it won't check your whole hard drive), and then if it didn't find the file it will give up and give you an error message. Since the procedure file is in the same directory as the main program, this will work well for us, no matter what directory they are both in, as long as they are together in the same directory.

When you write your own libraries, you may want to put the procedure file in a different directory than where the main program is. If you do this, you will want to put the whole path to the procedure file

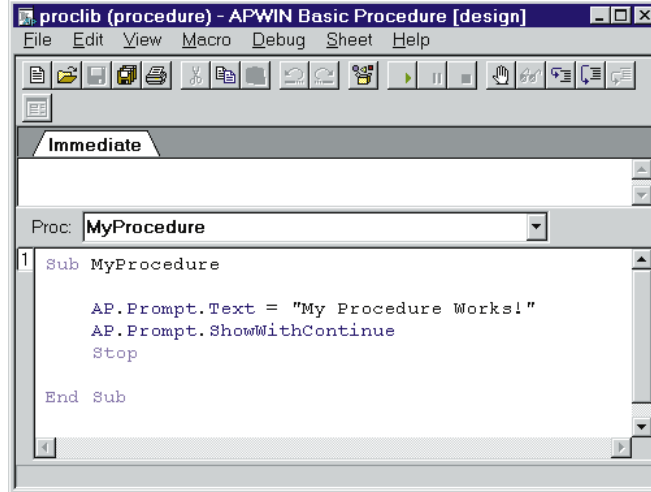
in the `uses` command. Otherwise, your program will not be able to find the library file.

You can have as many “`uses`” statements in your program as you want, telling your program about other files it might need, and use procedures from any or all of them. A word of caution, though, you do have to worry about two functions in different files having the same name. If the same name is used in two of the files you’re using, and you call that procedure by name, which will it use? I wouldn’t count on it using the one you wanted. Also, remember that `Main` is just another procedure, so if there is a `Main` procedure in one of your “`uses`” files, the language interpreter will give you an error message, saying you have two `Mains` so it doesn’t know which one to start with.

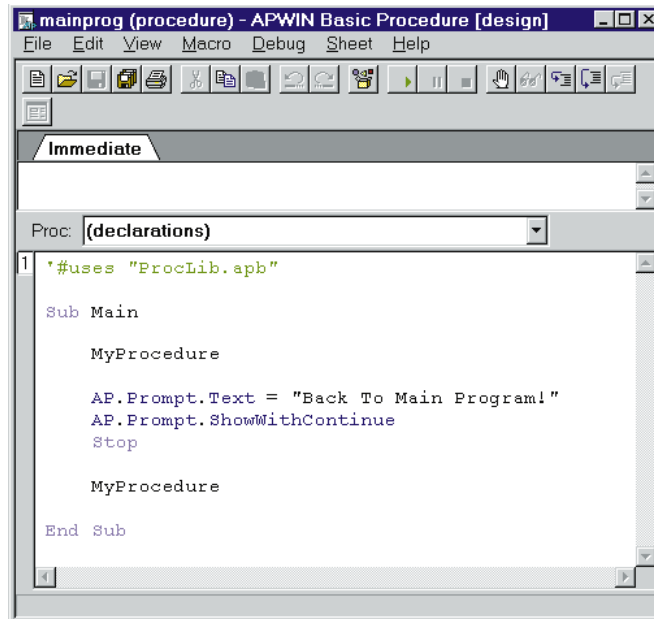
Here is an example of how to use the “`\#uses`” command. It is the same as the previous program, but it is broken up so that the procedure is in one file and the main part of the program is in another.

Type the `proclib` program in first, and then save it. Then start a new program and type in the `mainprog`. Then you can run `mainprog`, and it will call your procedure from the other file!

Of course, you can “cheat” and run `MAINPROG.APB` from the samples.



```
proclib (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help
[Toolbar]
Immediate
Proc: MyProcedure
1 Sub MyProcedure
    AP.Prompt.Text = "My Procedure Works!"
    AP.Prompt.ShowWithContinue
    Stop
End Sub
```

PROCLIB.APB Program

```
mainprog (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help
[Toolbar]
Immediate
Proc: (declarations)
1 '#uses "ProcLib.apb"
Sub Main
    MyProcedure
    AP.Prompt.Text = "Back To Main Program!"
    AP.Prompt.ShowWithContinue
    Stop
    MyProcedure
End Sub
```

MAINPROG.APB Program

4 Re-usable Program Pieces

Other Hints

Comments

Any good programmer will stress the importance of comments in your programs. Comments are just notes within your program which are meant to be ignored by the language interpreter. They are meant to be read by people who are looking at the program.

Sometimes it is not very obvious what the program is doing, because the APWIN Basic can be confusing when programs are complicated. Somebody else (or yourself!) may come back to the program some time later and have a lot of trouble figuring out what is going on. Embedding some notes in the program can be great help in making the program readable. You have already seen comments created by Learn Mode identifying when Learn Mode was started and stopped.

To add comments, use the single quote character ('). When the language interpreter finds this character, it will ignore the rest of the line, so you can put anything there that you want. Everything to the right of the quote mark is ignored completely when you run your program (with the exception of the "uses" command).

This is also a good way to skip a line that you don't want to use right now, but you don't want to erase. If you place a comment mark in front of it, the interpreter will ignore it, as though it wasn't even there. If you decide that you do want the line to be in the program, just erase the comment mark.

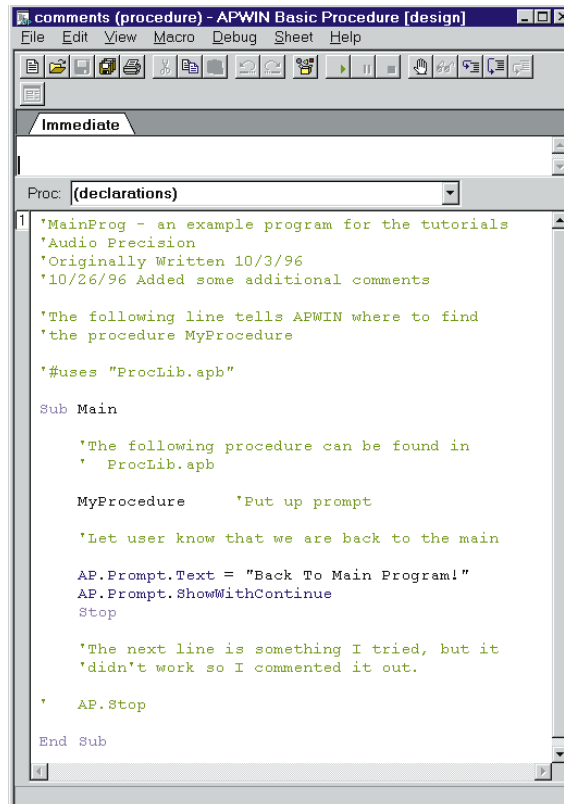
For example, let's say you want to run your program, but one test is causing you trouble. You can put single-quote marks at the beginning of all the lines that deal with the troublesome test. APWIN will treat these lines as comments, and ignore them completely. When you are ready to try the tests again, erase the quote marks, and your lines will be once again used as part of the program.

It is also a good idea to put some comments at the beginning that tells what your program does, who wrote it, and when and why any changes were made.

Don't hold any of the programs you have seen here as good examples of commenting (except the one below). Normally, they would all have

some comments, but these have been left out to make it easier to see what the actual language is doing. A good example of comments is the file Reports.apb, which is provided as a sample and is used in the next chapter.

Here is an example of a program with comments:



The screenshot shows a window titled "comments (procedure) - APWIN Basic Procedure [design]". The window has a menu bar with "File", "Edit", "View", "Macro", "Debug", "Sheet", and "Help". Below the menu bar is a toolbar with various icons. The main area of the window is a text editor displaying the following code:

```
1 'MainProg - an example program for the tutorials
  'Audio Precision
  'Originally Written 10/3/96
  '10/26/96 Added some additional comments

  'The following line tells APWIN where to find
  'the procedure MyProcedure

  '#uses "ProcLib.apb"

  Sub Main

    'The following procedure can be found in
    ' ProcLib.apb

    MyProcedure    'Put up prompt

    'Let user know that we are back to the main

    AP.Prompt.Text = "Back To Main Program!"
    AP.Prompt.ShowWithContinue
    Stop

    'The next line is something I tried, but it
    'didn't work so I commented it out.

    ' AP.Stop

  End Sub
```

A Program with Comments

Don't be afraid to experiment!

Before doing anything you are really unsure of, be sure to unplug your System One or System Two from any 'devices-under-test' that may be connected to its outputs, because it is possible for you to accidentally program the System to put out a fairly high voltage.

Beyond that, though, you can't break anything by playing around with programs. You can't hurt your computer, although we do recommend that you save data often. It is a good idea to make sure you have saved all important data on your hard disk before running experimental programs. In general, it is also a good idea to close down any other programs that are running on your computer.

You cannot break your System One or System Two by experimenting with programs, as long as you don't have anything connected to its connectors.

Creating Reports

Another common task for APWIN is to create a report of the results of a test or a group of tests. APWIN Basic includes powerful commands to allow it to interact with other programs like Microsoft Excel and Microsoft Word.

These programs have built-in languages of their own. Using a system called “OLE Automation”, certain programs can control each other, sending each other commands. In our examples, APWIN will be sending commands to Microsoft Word.

One of the most common requirements is to create a hard copy report showing test results. These test results can be in the form of graphs or numerical readings. We will be demonstrating how to insert both types of data into Word Documents.

This process is complicated, but we are going to avoid many of the complications by giving you procedures that will handle all the hard parts and exploring how to use these procedures. If you have read the section on using procedures (Re-usable Program Pieces), you should find it easy.

The procedures you need are found in the file Reports.apb. By default, this file will be installed in “C:\APWIN\Samples\Procedur”. If you chose not to install the samples, it was not installed, so you will have to run APWIN Setup again and install the samples.

In order for your program to be able to use these procedure, your program must have a ‘#uses’ command. This command (discussed in detail in the section on Re-usable Program Pieces) tells APWIN that your program will be using procedures from another file, and tells it where to find that file.

This process will also require a recent version of Word for Windows.

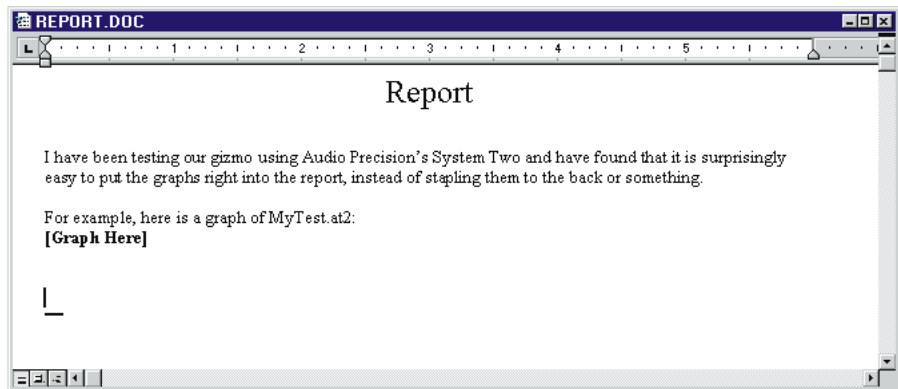
Including an Audio Precision Graph in your Reports

First we will take a graph from APWIN and insert it in a report that was written in Microsoft Word. This requires a procedure called `InsertGraphInReport` (which may be found in `Reports.apb`).

Before you use `InsertGraphInReport`, you need to create the report. But, instead of putting in the graph (which has not been created yet), you put in unique words to show where the graph goes. When you use the procedure, it will search for these unique words, and when it finds them, replace them with the graph.

It doesn't matter exactly what these words are, they just can't exist anywhere else in the document. When you call the procedure, you will be saying "Look in the document for these words and replace them with the graph". It will search through the document for the words you specified, and the **first** ones it finds will be replaced with the graph.

Using Microsoft Word, make a Word Document that looks like this.



(Hint: This file is provided with the samples!)

You can make your report different, but make sure the square brackets and the text inside them are exactly as you see here. Save your file in the directory with your programs.

After you have saved your report, close the report file (by selecting File Close from the menu) and minimize Word. Don't close the program entirely.

- If you are using Windows 3.1 or 3.11, hit the button in the top right corner of the Word window that looks like a down-pointing arrow. Word should reduce to an icon.
- If you are using Windows 95, look for the three buttons in the top right corner of the Word window. The minimize button is the leftmost of these three. When you hit this button, Word should shrink down to an item on the taskbar.

Then, go back to APWIN and enter the program, or load it from the samples. If you get a message that says "Can't open another sheet", then you have too many programs open and you need to close one or more. To close a program, press the right mouse button and select File Close from the menu

Each call to `InsertGraphInReport` is followed by three items, each surrounded by quote marks and separated by commas. These items tell `InsertGraphInReport` some specific details about how you want the job done. The items are:

1. The filename of the original report file (without the graph)
2. The text to find and to replace with the graph
3. What to do after inserting the graph

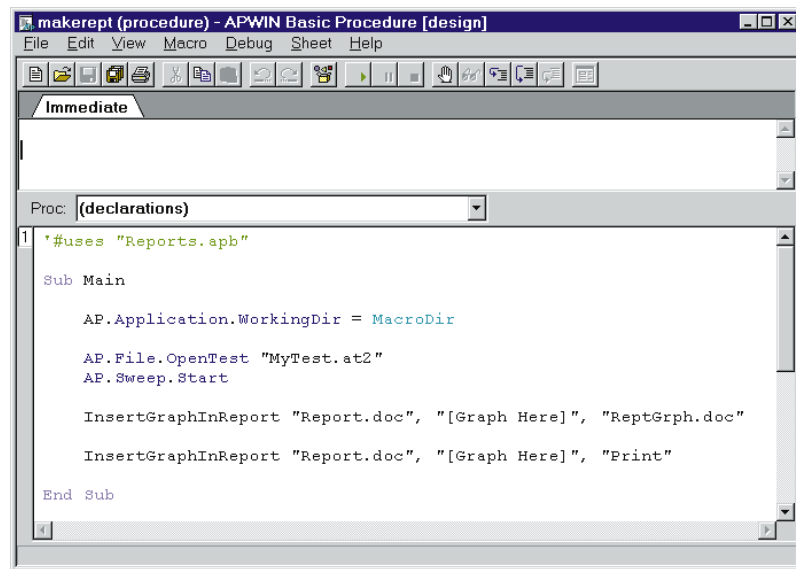
If the third item is "Print", then the report will be sent out to the default printer. If the third item is a filename, the report (with the graphs) will be saved to this file. Notice that we call the procedure twice - the first time, we save the report to a file, and the second time we send it to the printer.

Notice that there is a line in the program that asks for the report to be printed. If you do not have a printer, or do not want it to be printed, you have two options: you can either leave that line out entirely, or place a single quote mark at the beginning of the line. After adding the

quote mark, push the down arrow key to move to the next line down. When you do this, all the text on the entire line should turn green. This means that the line is 'commented out'. The quote mark tells the language interpreter to ignore this line, and run the program as though it doesn't exist. More about this in Chapter 5.

Also check the program to make sure it fits your situation - check that the filenames point to where the files are on your computer. If you are using System One, change the 'AT2' filename extension to 'AT1'.

Notice that the program has square brackets with words inside them which exactly match the square brackets and words in the report. If these do not match exactly, the program will not work correctly. Even the capitalization must be identical.



The screenshot shows a window titled 'makerept (procedure) - APWIN Basic Procedure [design]'. The window has a menu bar with 'File', 'Edit', 'View', 'Macro', 'Debug', 'Sheet', and 'Help'. Below the menu bar is a toolbar with various icons. The main area is divided into two panes. The top pane is labeled 'Immediate' and is empty. The bottom pane is labeled 'Proc: (declarations)' and contains the following code:

```

1  *#uses "Reports.apb"

Sub Main

    AP.Application.WorkingDir = MacroDir

    AP.File.OpenTest "MyTest.at2"
    AP.Sweep.Start

    InsertGraphInReport "Report.doc", "[Graph Here]", "ReptGrph.doc"

    InsertGraphInReport "Report.doc", "[Graph Here]", "Print"

End Sub

```

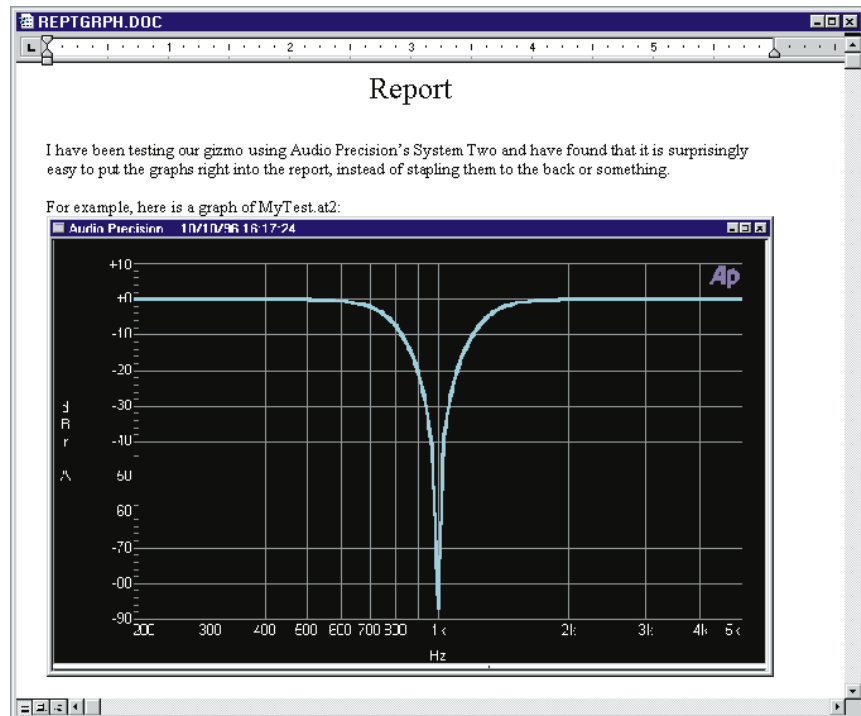
MAKEREPT.APB Program

Then **save** and run your program.

The program should run, and you should see it run the sweep in MYTEST.AT2. You may or may not see Microsoft Word and the report you entered. If you did not comment out the printing line, then the report should come out of your printer, with the graph in it.

While this is happening, you can bring Word up to the front (by clicking on its icon or taskbar item) and bring it up to the front, so that you can watch the program in action. This may be useful in exploring what your program is doing, but it will slow down the process considerably, since APWIN (which is running the show) will then be a background task. In normal operation, Word should be minimized and APWIN should be the selected application (it should be at the 'front', and its taskbar should be highlighted).

Then go back to Word and open the file "REPTGRPH.DOC". It should look like this:



If you load up original report (REPORT.DOC), it should not have the graph in it. Notice that the original report file was not changed. This means that you can use the same report file as many times as you want, inserting new graphs each time you use it. More about this later.

What if it didn't work?

As usual, the most common mistakes are typing errors. If APWIN gave you an error message, it will probably help you find the problem.

If your graph was inserted at the beginning of the report instead of where you wanted it, this means Microsoft Word did not find the text it was looking for, to tell it where to put the graph. If it doesn't find the text, it just puts the graph at the very beginning of the document. Make sure that the program's text (the part in square brackets) exactly matches the square-bracketed text in the report.

Also make sure all the paths and filenames you supplied are correct and valid. If you tell it to open a document and the document isn't there, it won't work correctly. And if you tell it to put a document in a certain directory and the directory doesn't exist, it won't know what to do.

If you left your report document open in Word, you will get a message from Word telling you that it can't save a document with the same filename as an open file.

If your report is created properly but doesn't print, you may have printing problems or no default printer. Go back to Word and try to print your report to the default printer.

If you got a different APWIN panel instead of the graph, it is probably because you selected that other panel with the mouse while the program was running. The procedure `InsertFileInReport` first selects the graph panel, then copies the selected panel into the clipboard for pasting to the Word document. If you selected a different panel before it did the copy, you will get that panel instead of the graph.

How did we do that?

In order to understand how this program works, we need to explain *parameters*. Parameters are just pieces of information you give a procedure when you call it. Parameters will tell the procedure some specific details about the job it has to do. Parameters are given directly after the procedure name, and the parameters are separated by commas.

The parameters that we use in this procedure tell it the name of the report document, and the specific words to search for to find where the graph belongs, and then what do do with the report after adding the graph. These must be in this exact order - there is no other way for the procedure to tell them apart. The last parameter can either be “Print” or it can be a filename. If it is “Print”, then the report will be sent out to the default printer. If it is a filename, the report will be saved with this filename.

As with tests and library files, if you don’t put in a path, `InsertGraphInReport` will expect to find the report files in the directory with your program. If you want to get your report from a different directory or save your report to a different directory, you can specify the entire path name instead of just the name of the report.

So, when we run this program, it starts reading at the “Sub Main”. The first instructions it reads tell it to load and run `MyTest` as we have seen in earlier examples. The next instruction tells it to run the procedure `InsertGraphInReport`. This instruction also tells `InsertGraphInReport` that the report to use is “REPORT.DOC”, that the text to look for is “[Graph Here]”, and when it has found the text and replaced it with the graph, to save the new report as “REPTGRPH.DOC”. APWIN Basic will run the procedure, and it will insert the current graph in place of the specified text and save it.

The next instruction tells APWIN to run the procedure `InsertGraphInReport` again. The main report file will be the same, and the text to find will be the same, but this time we tell it to send the report to the printer instead of saving it to a file. Since we have not rerun the sweep, the graph will still be the same. `InsertGraphInReport` will run again, replace the text with the graph, and print the report. Then the program will be finished.

As long as you understand how we are calling `InsertGraphInReport`, and how to supply the parameters, it is not too important that you understand how it works. You just need to know how to use it. Like a computer - do you know how a computer works? Some of you probably do, but the point is you don't have to know how it works, just how to get it to do the job.

Changing the Appearance of your Graph

The graph is copied directly from the screen image on the graph panel. Therefore, you will get an exact replica of what you see on the screen in your report. If you want the legend and comments on the bottom of the graph to appear with the graph in your report, adjust them to how you want them on the graph panel (see the APWIN User's Manual for how to do this) and save your test before running the program. If you want your traces to be thicker, you can also thicken them on the graph panel.

The page setup information is not used, since the graph comes directly from the screen image. Any changes you make in the page setup will not affect a graph in a report.

Going Further - More Graphs in a Report

Putting more graphs in a report is easy - all you have to do is put in new unique words for each of your graphs, and call `InsertGraphInReport` once for each graph you want to insert into your report.

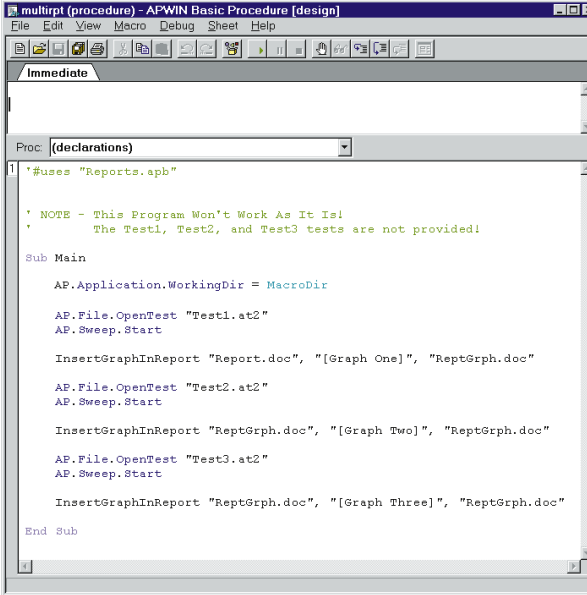
The only tricky thing about putting in multiple graphs is file management. Every time you insert a graph into your report, you have to tell `InsertGraphInReport` where to get the file and where to save it. If you don't plan these carefully, you will not get the results you expect.

Here is one case that won't work: say you want to put in two graphs, and print the file when you're done. If you call `InsertGraphInReport` twice, and each time, tell it to print the file out, you will get two reports printed, each with one graph and not the other! To get this to work properly, you need to have the first `InsertGraphInReport` save to a file. Then, the second time you call `InsertGraphInReport`, give it the name of this file as its source, and have it print when it's done.

As a general rule, when you are inserting multiple graphs in a report, you need to do the following: the first call to `InsertGraphInReport` should read from the original report file and write to another file, which we will call the work file. After the first, all calls to `InsertGraphInReport` should then read from the work file and write to the work file. The very last call should read from the work file and then either save it to the place you want it permanently saved, or send it out to the printer.

On the following page is an example of a program that puts several graphs in the same report. In order to use this program, the report file should have places for three graphs, marked with “[Graph One]”, “[Graph Two]”, and “[Graph Three]”. There are more examples of this technique in the following section.

Note: This program won't work as it is! The tests are not provided. To use it, change the test names to some of your own tests.



```

multirpt (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help
Immediate
Proc: (declarations)
1 '#uses "Reports.apb"
' NOTE - This Program Won't Work As It Is!
'   The Test1, Test2, and Test3 tests are not provided!
Sub Main
  AP.Application.WorkingDir = MacroDir
  AP.File.OpenTest "Test1.at2"
  AP.Sweep.Start
  InsertGraphInReport "Report.doc", "[Graph One]", "ReptGrph.doc"
  AP.File.OpenTest "Test2.at2"
  AP.Sweep.Start
  InsertGraphInReport "ReptGrph.doc", "[Graph Two]", "ReptGrph.doc"
  AP.File.OpenTest "Test3.at2"
  AP.Sweep.Start
  InsertGraphInReport "ReptGrph.doc", "[Graph Three]", "ReptGrph.doc"
End Sub

```

MULTIRPT.APB Program

Including Numerical Data in your Reports

You can also include numerical readings in your reports in a similar manner. You may either include a single reading or an entire sweep of readings.

We will be using a procedure called “InsertDataInReport”. It behaves similarly to InsertGraphInReport. When you call it, you will tell it what words to look for, and it will place the reading or readings in place of these words. You should create the report first, inserting the unique words everywhere you want a reading or a number of readings to be placed.

Your procedure should load and run a test before using the `InsertGraphInReport` function to obtain the desired data. Your sweep should contain the same number of points that you want inserted in your document.

If you only want one data point, it may seem silly to set up a whole sweep just for one point. However, there are several advantages to doing this, rather than getting the reading directly from the panel. The first is that it allows you to use the same procedure for a single point as for a large number of points. The second is that it allows you to use the regular Sweep Settling panel to define your settling parameters. If your program was trying to read the reading directly from the panel, you would have to write your own settling, and you don't want to do that.

The `InsertDataInReport` handles data one column at a time. The columns are the same as shown on APWIN's Data Editor. Each column is a series of data points. If you run a sweep that uses one source and makes two measurements, you will have three columns - one with every point that the source used, one with every point taken by the first measurement, and one with every point taken by the second measurement.

For each column of data you want to insert in your report, you need a unique word where first reading of the column would be. You can create a single table in your report and then insert columns of data into it from a number of different sweeps. For each column you insert, you will call `InsertDataInReport`.

Usually, if you are inserting a number of data points in your report, you will want to put your data in a table. If you have not used Microsoft Word's table features before, you may want to take a look at them now. Placing the data in a table gives you control over columns of data, keeping the columns lined up and allowing you to change the column widths. While you could get away with not using a table for a single column of data, it is definitely a good idea to use a table if you are using more than one column - for example, if you use one column for the generated frequencies and another for the measured data.

If you do not use a table, you have to be very careful about where the program is going to insert the data. After each data point is inserted,

the program will instruct Word to ‘move the cursor down one’, just the same as if you had hit the down arrow key. In a table, this will always move to the next cell down, but the behavior can be quite different if you are in a block of text. Each down arrow moves down a line, but also moves over to wherever the cursor was last positioned on that line.

Note that `InsertDataInReport` does not actually insert lines or graph rows, it only adds to lines already there. Therefore, you need to make sure that there are sufficient lines or cells for all the data you wish to insert.

If you are only inserting one data point, you can put it anywhere, even in the middle of a sentence.

The text attributes of the data (font, size, etc.) are not specified by the program - they must be in the document. The first reading (or the only reading, in the single-point case) will have the same attributes as the text it replaced - the word it was searching for. If you are inserting a whole column of data, and you use a table, it is best to highlight the entire table and set the font, size, and any other attributes you want. Word will then assign these attributes to the individual cells, and the data will be formatted to match.

`InsertDataInReport` requires seven parameters. The first three are the same as the ones used for `InsertGraphInReport`. You must use all seven, in the correct order, or your program will not work correctly:

1. The filename of the report file to load.
2. The unique words to find in the document, showing where to put the data.
3. The filename to save to, or the word “Print”.
4. The number of the column of data to insert in the report.
5. The number ‘1’ to add normal suffixes to numbers (k, m, u, etc.), 0 to leave them as they are.
6. The units to use for the readings.
7. The format of the readings.

The first parameter specifies which report to load. It should be enclosed in double-quote marks. If you don't provide the entire path name, APWIN will only look for the report file in the directory of the program. If it isn't there, you will get an error.

The second parameter, also enclosed in double-quotes, should contain the unique word or words to find. If you are using a table, the words should be *in* the table, at the top of where you want your data to be. Remember, the first data point will be placed directly on top of the unique words. If the words are not found, the data will be inserted at the beginning of the document.

The third parameter should be the filename to save the report to, after adding the data to it. If you don't provide the entire path name, APWIN will save the report in the directory of the program. Or, you may use the word "Print", which will cause the file to be sent to the default printer instead of being saved. Whichever of these you choose, the parameter should be enclosed in double-quote marks.

The fourth parameter is the column number to insert in the report. The columns are in the same order as shown on APWIN's data editor. Column 0 is the source data, column 1 corresponds to Data 1, column 2 to Data 2, etc. If you ask for a column number that is larger than the number of columns of data available, the results will be unpredictable. This parameter should *not* be enclosed in quote marks - in fact, it should not be enclosed in anything, just a naked number.

The fifth parameter is also a naked number, but this time only a 1 or 0. If the number is 1, the usual suffixes will be added to make the data more readable. For example the number "23400 Hz" will be modified to read "23.4 kHz", and so forth. The following suffixes will be added, if appropriate: G, M, k, m, u, n, p. If the parameter is 0, the data will be given in a number-only format, without any suffixes added.

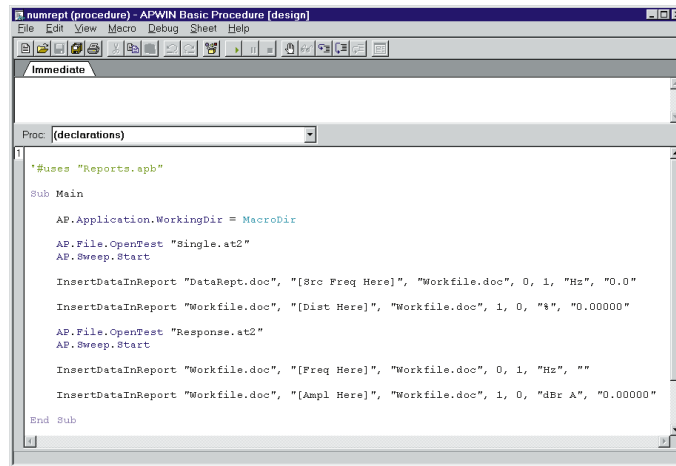
The sixth parameter is the units of the reading, enclosed in double-quotes. This does not need to be the same units as you used in your sweep, although they must be compatible units. In fact, you can put the same data in your report in two different units. For example, you might have a table in your report that lists your reading in volts and also dBm. To do this, call InsertDataInReport twice, the first time asking for the reading in volts, the second time asking for

dBm. APWIN will perform the necessary conversions. To find out which units are compatible with a given reading, go to the APWIN panel with that reading and pull down the drop-down box of the reading. It will show all the units that are available for that reading.

The seventh parameter is the format of the data, also enclosed in double quotes. There are many options for this format parameter, but the most common is to specify the number of digits you want before and after the decimal point. To do this, the format parameter should be a few 0's, and then a decimal point, and then a few more zeroes. The meaning of the zeroes before the decimal point is slightly different than the meaning of the zeroes after the decimal point. The zeroes before the decimal point mean 'put at least this many numbers here, adding zeroes in front if necessary'. The zeroes after the decimal point mean 'put only this many numbers in'. Normally, you would use only one number before the decimal point, and as many after the decimal point as you want for the desired accuracy.

If you don't want your output formatted (or just want to see what happens!) you can omit this parameter, but you must put in the quotes. This means that the parameter will just be two double-quotes, with nothing (not even a space) in between, like this: `""`. This is the way of saying nothing.

Here is an example program to run with the report.



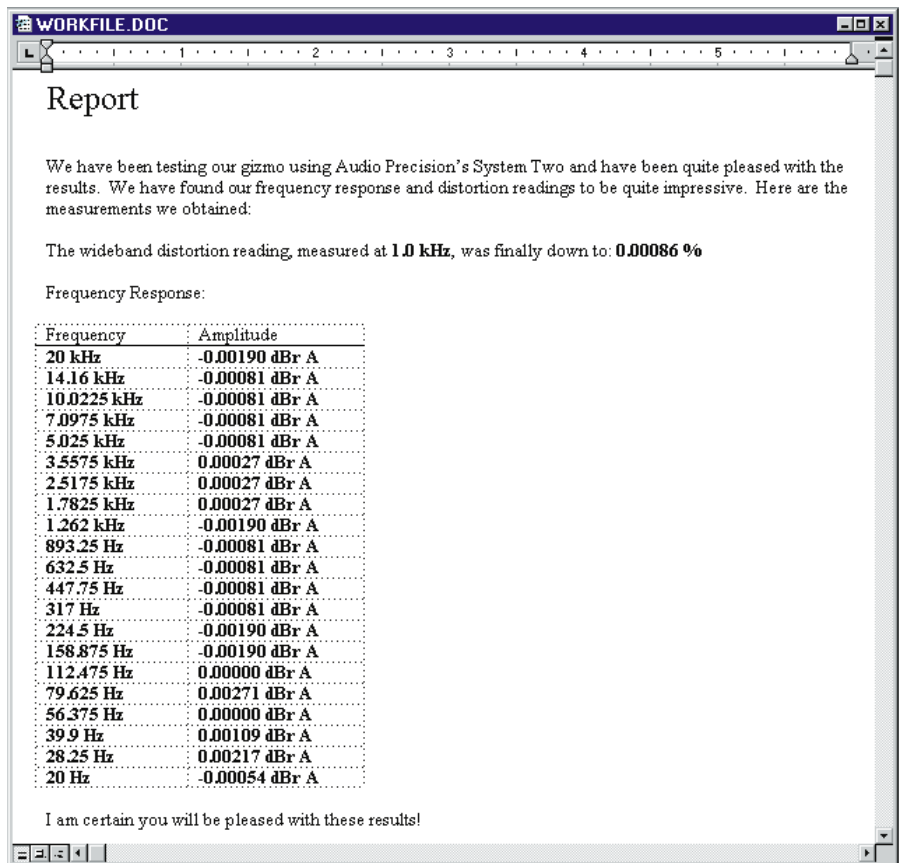
```
numrept (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help
Immediate
Proc: (declarations)
1
"Uses "Reports.apb"
Sub Main
  AP.Application.WorkingDir = MacroDir
  AP.File.OpenTest "Single.at2"
  AP.Sweep.Start
  InsertDataInReport "DataRept.doc", "[Src Freq Here]", "Workfile.doc", 0, 1, "Hz", "0.0"
  InsertDataInReport "Workfile.doc", "[Dist Here]", "Workfile.doc", 1, 0, "%", "0.00000"
  AP.File.OpenTest "Response.at2"
  AP.Sweep.Start
  InsertDataInReport "Workfile.doc", "[Freq Here]", "Workfile.doc", 0, 1, "Hz", ""
  InsertDataInReport "Workfile.doc", "[Ampl Here]", "Workfile.doc", 1, 0, "dBc A", "0.00000"
End Sub
```

NUMREPT.APB Program

This program refers to two tests - Response.at2, which runs a 20-step (21 total points!) response sweep, and Single.at2, which runs a single point distortion sweep. Check filenames, directories, and filename extensions to make sure they match your situation.

System One Users: In addition to the usual test name changes, you need to change the “dBr A” to “dBr” - in System One, there is only one dBr reference, so there is no “dBr A” and “dBr B”, only “dBr”. If you forget to make this change, APWIN will give you an error message telling you that the units are invalid.

After running this program, take a look at the Workfile document. It should look like this (except, of course, the actual data will be different):



Often one program can be written to perform several slightly different tasks. For example, maybe you are testing two types of instruments that are similar but not identical. You could write a different program to test each of the instruments, or you could write one program which would put up a menu and let you choose which instrument you wanted to test. This section covers the programming of these menus.

Menus can also help you streamline your work. If you have several different programs that you regularly run, you can set up a menu to run them for you. Then, you won't have to go through the process of loading and running a different program every time you change tasks. You just run your menu program, and press the button for whatever program you want to run. You can even write your program so that it will come back to the menu when it finishes. That way, you are ready to run the next program with a single keypress.

In order to make this a little less complicated, we need another term: the *user*. The user is the person who presses the buttons on your menu. This might be you, or it might be someone that you give the program to. As the programmer, you will be creating menus for the user to use.

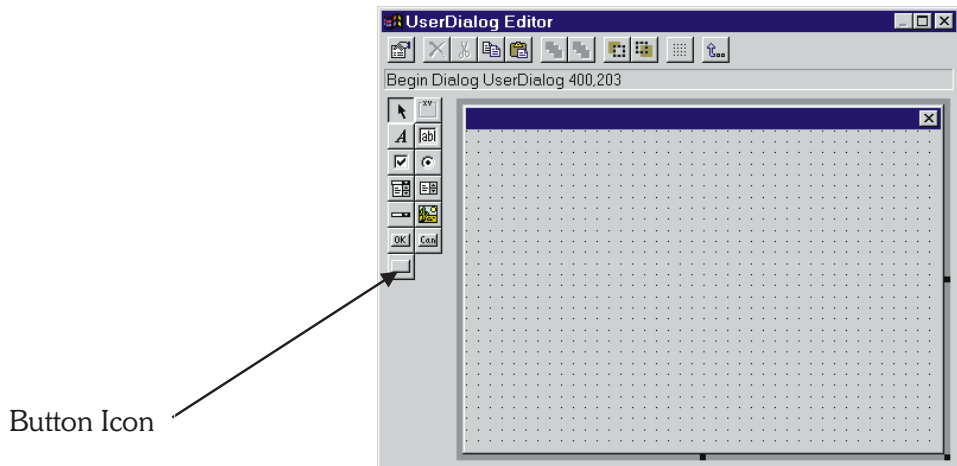
When a program puts up a window to get some information or directions from the user, we call this window a *dialog box*. You have probably seen dialog boxes before - an example is the window that you get from APWIN when you select the menu item Utilities Configuration. It will give you a number of check boxes that allow you to define some things about the way APWIN runs. Another example is what you get from File Open - this dialog box is designed to help you select a file to open.

APWIN Basic can create very sophisticated dialog boxes, with all sorts of controls and buttons like you see in many Windows programs. We're only going to discuss the most useful ones, which you should find fairly easy to use.

The User Dialog Editor

First, let's get introduced to a very useful feature of APWIN - the UserDialog Editor. This feature creates dialog boxes for you. You can place buttons and controls wherever you want in your dialog box, and move them around by dragging them with your mouse. You can also add text instructions for the user in your dialog box. When you get your dialog box looking the way you want it, APWIN will write the dialog box instructions for you. It will insert these instructions in your program, and when you run your program, you will see your dialog box just the way you designed it, and the user will be able to push the buttons and give information to your program.

Start with a fresh, clean program so that we can tell what APWIN is writing. Open up your procedure editor and get a clean page (by pressing the 'New Procedure' button). Put your cursor on the empty lines between Sub Main and End Sub, and press the right-hand mouse button. You will get a menu, from which you select Edit - UserDialog. This window should pop up:



This window is actually one window inside another. The inside one, which is filled with nothing but little dots, is your dialog box, where you can put whatever you want (within reason). If you want to change the size of your dialog box, you can go to the little black squares that are in the bottom right corner and along the bottom and right edges and drag them to whatever size you like. You may need to increase the size of the Dialog Editor window (by dragging its bottom right corner) so that you can see all of your dialog box.

The icons along the left side are used to put new things into your dialog box. To add something to your dialog box, press one of the icons along the left. Then, put your mouse pointer in your dialog box and click wherever you want the item. The first one we will be using is the “Button icon”, marked on the picture above.

You can move your new item by going to the middle of it and holding down your left mouse button. While holding the button down, you can move the item to wherever you want. If you want to change the size of the item, you can drag the little black squares that are on the edges.

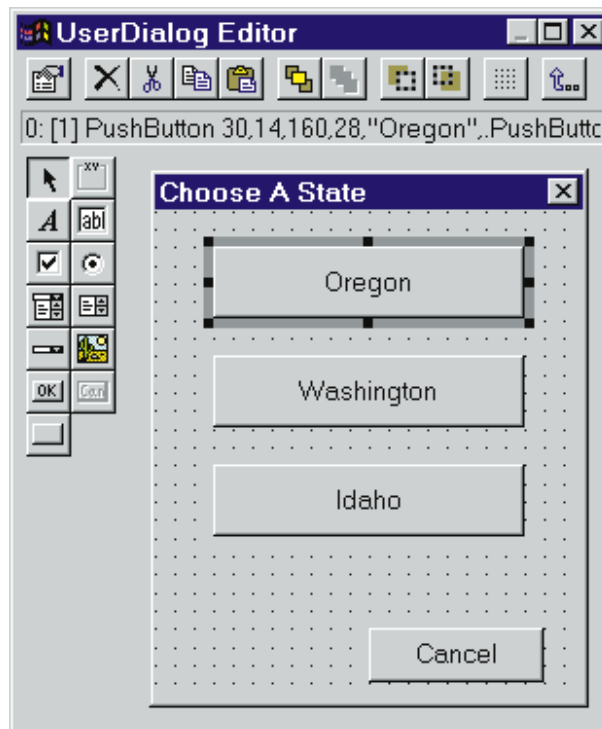
You can play with this if you want - put a few things in your dialog box and move them around a little. You can only resize and move an item if it is ‘selected’. You can select any item by clicking on it, and you should see a gray outline around it. To delete the item, hit the scissors icon.

If you double-click on any item in your dialog box, you will get a window which will show some details about the item. Most of these you won’t need to worry about, but you will want to change the one marked ‘Caption’ - this shows the text that appears on the item. You can also double-click on the title bar (the bar along the top) of your dialog box to change the caption that appears in the title bar.

To add some instructions for the user, press the icon that looks like a capital A, and then click inside your dialog box. This will let you add an item that doesn’t do anything, but it has a caption, so you can add some instructions for the user.

A Simple Menu

When you are ready to go on with the example, delete all the items from your dialog box. Then press the “Button icon”, marked on the picture above, and use it to add three buttons to your dialog box. Be sure to start with the top button, then the middle, then the bottom button. Then, change the caption of the top button to “Oregon”, the middle one to “Washington” and the bottom one to “Idaho”. Also, change the title bar to “Choose A State”. The window should look like this:

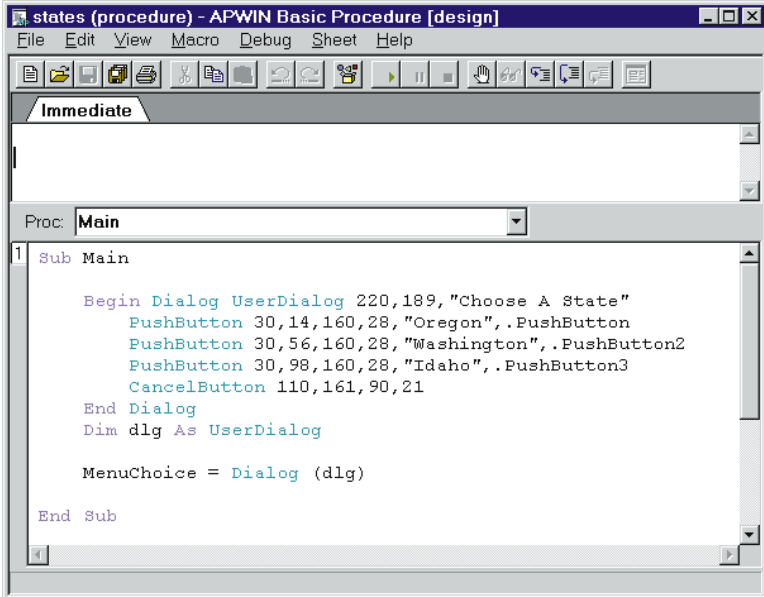


It doesn't really matter that your windows or buttons are the same size as these ones, only that you have the buttons in the right order.

There's one very important thing we need to learn from the UserDialog Editor before we leave: the “button number” of each of our buttons. To find out the number of each of your buttons, select the button (by clicking once on it) and look up near the top of the UserDialog Editor.

Right under the top toolbar is a line of text with some numbers, words, and commas which shows everything about the button. The important number is in square brackets over toward the left. This number is all we need to know for our program to use this button. Check each of the buttons to make sure that Oregon is [1], Washington is [2], and Idaho is [3]. If they are not correct, the only way to fix them is to erase all your items and start over, because they are assigned in the order that they are created.

When your dialog box is finished, press the X up in the top right hand corner (of the UserDialog Editor, not your new dialog box!). It will ask you if you want to save your work. Press Yes. You will then be delivered back to the editor, where you should see that your program has been modified - mine looks like this:



```
states (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help
[Toolbar]
Immediate
Proc: Main
1 Sub Main
  Begin Dialog UserDialog 220,189,"Choose A State"
    PushButton 30,14,160,28,"Oregon",.PushButton
    PushButton 30,56,160,28,"Washington",.PushButton2
    PushButton 30,98,160,28,"Idaho",.PushButton3
    CancelButton 110,161,90,21
  End Dialog
  Dim dlg As UserDialog
  MenuChoice = Dialog (dlg)
End Sub
```

STATES.APB Program

Run your program. You should see your dialog box, which will wait until the user pushes one of the state buttons, and then end. We haven't yet given the program anything to do in response to button-pushes.

You don't have to worry much about what these instructions are or what they do. APWIN will take care of that. After all, you didn't write it, why should you be responsible for it? If you want to change your dialog box, put your cursor somewhere between "Begin Dialog" and "End Dialog" and then start the UserDialog Editor. It will automatically detect the fact that you are editing an existing dialog box and let you make changes to it. If you are anywhere outside of it, APWIN will start a new dialog box for you to edit.

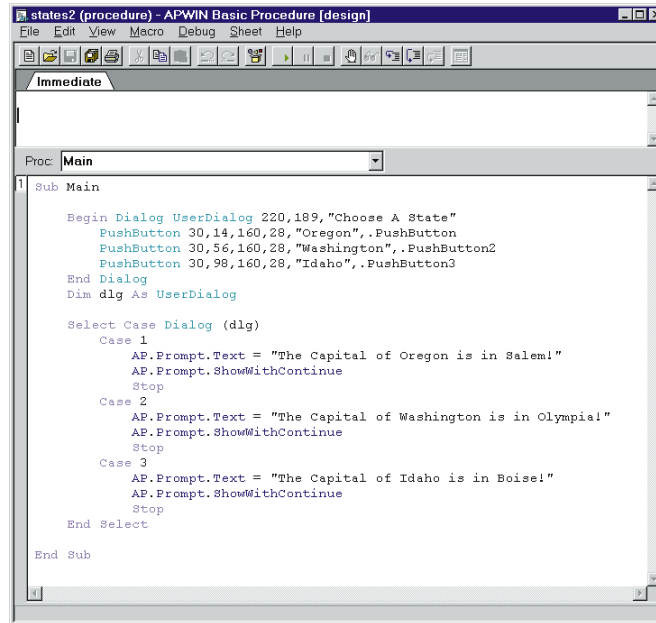
The only line we will have to change is the last one that APWIN wrote, which says "Dialog dlg". Everything before that line is defining what the dialog box will look like and how it will behave. This is the line that says "Ok, now put up that dialog box I told you about". If you want to, add another line below that line that is an exact duplicate of it and run the program. This time, you should see the dialog box twice.

Then erase the duplicate line so that our programs are the same again.

These commands which APWIN wrote completely define your dialog box. If you want to use the same dialog box again, you can copy this block of instructions and you will get an identical dialog box, with all the same properties.

Now modify your program so that it looks like the code shown in the editor on the following page (or load STATES2.APB from the samples).

Save your program and run it. You should be given a menu of states to pick from, and when you pick one you should get a window telling you what the capital is. If you get a different state from the one you chose from the menu, then your button numbers are wrong. You will need to go back and edit your dialog box so that Oregon is [1], Washington is [2], and Idaho is [3].



```

states2 (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help
-----
Immediate
-----
Proc Main
1 Sub Main
  Begin Dialog UserDialog 220,189,"Choose A State"
    PushButton 30,14,160,28,"Oregon",.PushButton
    PushButton 30,56,160,28,"Washington",.PushButton2
    PushButton 30,98,160,28,"Idaho",.PushButton3
  End Dialog
  Dim dlg As UserDialog

  Select Case Dialog (dlg)
    Case 1
      AP.Prompt.Text = "The Capital of Oregon is in Salem!"
      AP.Prompt.ShowWithContinue
      Stop
    Case 2
      AP.Prompt.Text = "The Capital of Washington is in Olympia!"
      AP.Prompt.ShowWithContinue
      Stop
    Case 3
      AP.Prompt.Text = "The Capital of Idaho is in Boise!"
      AP.Prompt.ShowWithContinue
      Stop
  End Select

End Sub

```

STATES2.APB Program

How did we do that?

First, I need to introduce a new language feature, which is called `Select...Case`. This is used to choose between different sections of your program. When your program is running, it will decide which piece of the program to run, based on a number that you give it.

Normally, a `Select...Case` statement would look like this:

```

Select Case <number>
  Case 1
    <some instructions for Case 1>
  Case 2
    <some instructions for Case 2>
  Case 3
    <some instructions for Case 3>
End Select

```

The things in *italics* would not look exactly like you see them here. Where it says <number>, you would just put a regular number like 1 or 2 or whatever. Where it says <some instructions for...> would be some regular APWIN Basic instructions. When the language interpreter sees a *Select...Case* statement, it looks at what the number is, and then goes down and follows the instructions under that *Case*. For example, if the number was 2, it would only run the instructions for *Case 2*, and then it would jump down to where it says “*End Select*” and start reading from there.

If you look at the program shown above, you will notice that we don't give it a number - instead it says “*Dialog (dlg)*”. This tells APWIN to put up your dialog box, let you push a button, and give back the number of the button the user pushed. This number will be given to *Select...Case* statement, which will then jump to whatever piece of program is shown for that number.

You can put any number of instructions for any *Case*, but if you have too many instructions, so that you can't see the whole *Select...Case* statement all at once, it can get confusing. The best way to deal with this is to make a procedure out of the instructions you want to run, and then just have the *Select...Case* statement call the procedure (see the section on Re-usable Program Pieces to see how to do this). This way, your *Select...Case* statement is kept simple and readable.

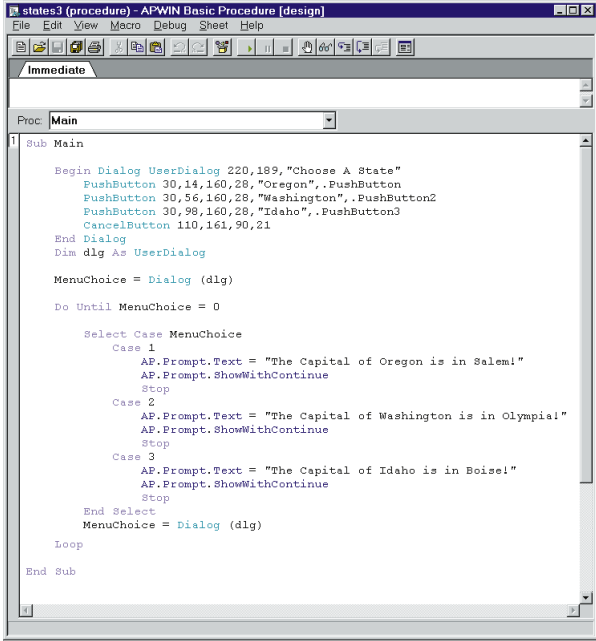
Going Further - Restarting the Menu each time

The next thing we will do is change the program so that after giving us the name of a capital, it will start the menu over again and let the user choose another capital.

The first thing we need to do is add a *Cancel* button to our dialog box. If we didn't do this, we would have no way to tell the program that we were finished. We could choose all the capitals we wanted, but when we had seen them all we would probably want to stop and do something else. Any button we pushed would just show us another capital and bring us back to the menu. Put your cursor between the ‘*Begin Dialog*’ and ‘*End Dialog*’ buttons and use the menu to select *Edit - UserDialog*, starting the *UserDialog* Editor.

You could just add another button, which would be button [4], and call it cancel, but the UserDialog Editor provides a special cancel button. It is down below the “Button icon” that we used before, and says “Can” in it. Click on this icon to select this type of item, and then click where you want your cancel button. If you look up at the top of the window where the button number normally is, you will notice that it doesn’t have one. The cancel button is always button number [0]. You can only have one Cancel button in each dialog.

After adding your cancel button, go back to your program. You should see that the instructions APWIN wrote to create your dialog box have changed - it has added instructions for the Cancel button. Then change your program so that it looks like this: (or load the sample)



```
states3 (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help
Immediate
Proc: Main
Sub Main
  Begin Dialog UserDialog 220,189,"Choose A State"
    PushButton 30,14,160,28,"Oregon",.PushButton
    PushButton 30,56,160,28,"Washington",.PushButton2
    PushButton 30,98,160,28,"Idaho",.PushButton3
    CancelButton 110,161,90,21
  End Dialog
  Dim dlg As UserDialog

  MenuChoice = Dialog (dlg)

  Do Until MenuChoice = 0
    Select Case MenuChoice
      Case 1
        AP.Prompt.Text = "The Capital of Oregon is in Salem!"
        AP.Prompt.ShowWithContinue
        Stop
      Case 2
        AP.Prompt.Text = "The Capital of Washington is in Olympia!"
        AP.Prompt.ShowWithContinue
        Stop
      Case 3
        AP.Prompt.Text = "The Capital of Idaho is in Boise!"
        AP.Prompt.ShowWithContinue
        Stop
    End Select
    MenuChoice = Dialog (dlg)
  Loop
End Sub
```

STATES3.APB Program

Save your program and run it. You should see the same menu as we saw before, and it should work the same as it did before, except that after seeing each capital you will return to the menu. Then, if you press Cancel, your program will end.

How did we do that?

We need two new language features to explain this program. The first feature is called a variable. A variable can be thought of as a box, somewhere inside the computer, which is given a name. Your program can then put anything it wants into that box. The item will stay in the box until you put something else in there, but you can only have one thing in the box at a time. Any time your program wants to know what's in the box, it can ask for it by name. This is usually used to store something that you need later.

Normally you will put a number into a variable using the equals sign. For example, if your hat size is 12, you could store the number 12 in MyHatSize like this:

```
MyHatSize = 12
```

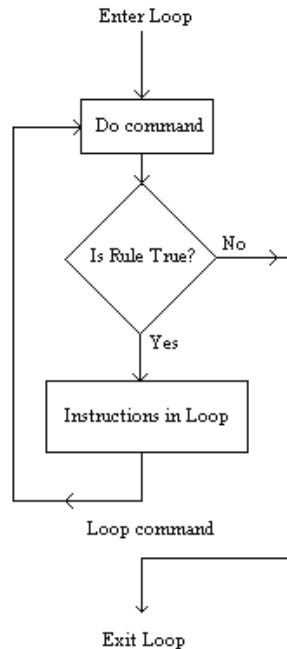
This creates a variable called MyHatSize and stores the number 12 in it. Then, if (for some reason) your program need to use your hat size in a calculation, it will just call it by the name MyHatSize. For example, if your program calculated "MyHatSize + 4", the result would be 16.

The variable in this program is called MenuChoice. The first time you see the variable name in the program is in the line that says "MenuChoice = Dialog (dlg)". This tells the program to put up the dialog box and let the user pick from the menu. It will then take the button number of the button you pushed and put this number into the variable MenuChoice. From then on, any time we want to know what button was pushed, we just look at MenuChoice. For example, take a look at the Select statement. In the place where a number usually belongs, we have the word MenuChoice. This tells the Select...Case to use the number stored in that variable to decide which instructions to execute.

The second new language feature that makes this program work is the Do...Loop feature. These commands are used to make your program repeat a group of instructions several times.

Right after the word ‘Do’ is the loop rule, which determines how many times to go through the loop. The loop will be repeated until the loop rule is no longer true. The word ‘Loop’ defines the end of the loop.

The sequence of a Do...Loop is like this:



The Do command and the Loop command don't really perform any action, they just mark the beginning and the end of the loop. Notice that the rule will be checked before even running the instructions in the loop once. This means that the instructions may never be run at all if the rule decides to stop the loop on the first pass.

Let's go through the program step-by-step so we can see how the loop works.

The first block of instructions defines the dialog box. Then we reach the line “MenuChoice = Dialog(dlg)”, which tells APWIN to put up the dialog box. Assume the user presses the Washington button, so the number 3 is assigned to the variable MenuChoice.

Then we see the Do instruction, which is followed by the loop rule. The rule says “continue the loop until the variable MenuChoice has a value of zero”. Since MenuChoice is 3, not 0, we run the instructions in the loop.

Inside the loop is the Select...Case statement, which uses the MenuChoice variable to decide which part of the program to go through. Since MenuChoice is 3, it will run the instructions that display the capital of Washington.

After the Select...Case is finished, the next instruction is “MenuChoice = Dialog(dlg)” again. This puts up the menu again, allowing the user to make another choice. Assume the user chooses Idaho.

The next instruction we see is Loop. This means we’ve reached the end of the Do...Loop. When the language interpreter reaches this instruction, it will go back to the Do instruction and check the rule to see if it needs to go through the loop again. Since MenuChoice is still not 0, the instructions inside the loop will be repeated. The Case statement will be evaluated again, and the capital of Idaho will be shown.

Then we reach the Dialog statement again. This time, assume the user presses Cancel. The value of 0 will be stored in MenuChoice.

Next we reach the Loop instruction again, so we go back up to the Do and evaluate the loop rule again. Since MenuChoice is now 0, the rule says that the loop is finished, so we jump down to the instruction after Loop. The next instruction is End Sub, so the program is finished.

It may seem a little bit confusing that we put the menu choice at the end of the loop. We do this because the rule check (that determines whether we should go through the loop again) always happens at the beginning of the loop, and we want the rule check to be right after the menu choice. After all, if you chose Cancel, why should we go through the Select...Case statement? One good way to think of a loop is as a circle, with the rule check at the very top of the circle. If you want something to happen right before the rule check, it needs to be the last thing on the way around the circle.

Whenever you make a looping structure, be very careful to provide a way out. This is even more important when working with menus,

because menus quite often disable APWIN's stop button. If your program is caught in a loop that continuously puts up menus, you might not be able to stop it. **ALWAYS SAVE YOUR WORK** before running a looping program - certain errors may require you to restart APWIN or possibly even your whole computer. If you do get stuck in a loop, first try the stop button. If that doesn't work, use the keyboard to keep your program rapidly going through the loop, and click your mouse continuously on the stop button. If you are lucky, you will hit the button at just the right second and the program will stop. If that doesn't work, you will need to try Ctrl-Alt-Del, which will abort APWIN, or restart your computer.

Multiple Menus in One Program

Putting more than one menu in a program gets complicated, but you can avoid all the problems by using one simple technique. The trick is to put only one menu in any procedure. If you skipped over the section on Re-usable Program Pieces, you may want to look back at that section now.

Why this trick works is too complicated to explain in this section (it has to do with the scope of variables), but it works very well. It is possible to put more than one menu in a certain procedure (remember that 'Main', where we have been doing most of our work, is just another procedure), but it is just not worth the hassle. Things go so much more smoothly if you just make a new procedure and put your menu in there. If you do this, all of your menus can look exactly like I have shown in the above examples (except that you will want to change the dialog box and the instructions for each Case). Other than that, you can just copy all the instructions.

Here is an example of a program that uses multiple menus. Notice the similarity between the Main Procedure and the DigitalMenu Procedure. LastChanceMenu is different, because it doesn't repeat - it goes up once, and then the program ends.

Load MultMenu.apb from the samples and run it. The first menu will let you choose Analog Tests or Digital tests. Selecting Digital Tests will give you the Digital Menu. Selecting quit from the Digital Menu will take you back to the Main Menu. Selecting quit from the Main Menu will take you to the Last Chance menu. You will only see the Last Chance menu once - after you have made your selection once, the program will end.

Run this program a few times and try the various options. A good way to see what the program is doing is to print it out, and then follow the program on your printout as you press buttons on your menus. Each Dialog command will give you a menu, and then you can pick one of the options and watch the Case statement to see what the program will do, and where it sends you. Remember that whenever a procedure ends, the language interpreter will go back to where the procedure was called from and restart right where it left off, almost as if the procedure had not been there.

```

multimenu (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help
-----
Immediate
Proc: Main
1 Sub Main
  Begin Dialog UserDialog 220,140,"Choose Type of Test"
    PushButton 30,14,160,28,"Digital Tests",.PushButton
    PushButton 30,56,160,28,"Analog Tests",.PushButton2
    CancelButton 120,112,90,21
  End Dialog
  Dim dlg As UserDialog

  MenuChoice = Dialog (dlg)
  Do Until MenuChoice = 0
    Select Case MenuChoice
      Case 1
        DigitalMenu
      Case 2
        AP.Prompt.Text = "Analog Menu Not Yet Created!"
        AP.Prompt.ShowWithContinue
        Stop
    End Select
    MenuChoice = Dialog (dlg)
  Loop
  LastChanceMenu
End Sub

Sub DigitalMenu
  Begin Dialog UserDialog 220,140,"Choose Type of Test"
    PushButton 30,14,160,28,"FFT Test",.PushButton
    PushButton 30,56,160,28,"MLS Test",.PushButton2
    CancelButton 100,105,90,21
  End Dialog
  Dim dlg As UserDialog

  MenuChoice = Dialog (dlg)
  Do Until MenuChoice = 0
    Select Case MenuChoice
      Case 1
        AP.Prompt.Text = "FFT Test Not Yet Created!"
        AP.Prompt.ShowWithContinue
        Stop
      Case 2
        AP.Prompt.Text = "MLS Test Not Yet Created!"
        AP.Prompt.ShowWithContinue
        Stop
    End Select
    MenuChoice = Dialog (dlg)
  Loop
End Sub

Sub LastChanceMenu
  Begin Dialog UserDialog 220,140,"Last Chance To Choose!"
    PushButton 30,14,160,28,"Good Choice",.PushButton
    PushButton 30,56,160,28,"Fick Me",.PushButton2
    CancelButton 100,105,90,21
  End Dialog
  Dim dlg As UserDialog

  Select Case Dialog (dlg)
    Case 1
      AP.Prompt.Text = "Actually, the other choice was better."
      AP.Prompt.ShowWithContinue
      Stop
    Case 2
      AP.Prompt.Text = "This isn't exciting. Try the other one."
      AP.Prompt.ShowWithContinue
      Stop
  End Select
End Sub

```

MultMenu.APB Program

Using Check Boxes in your Dialog Boxes

Another useful dialog box feature is the check box. Check boxes are what you see in the Utilities - Configuration dialog box - a little white box which can be checked or unchecked. You can add check boxes to your dialog boxes, which will give the user a chance to choose some different options about how your program is run.

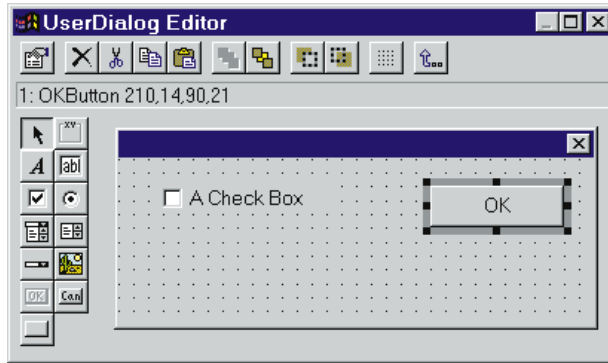
You can set up these check boxes to mean whatever you want. After your program puts up your dialog box, and the user checks or unchecks any desired boxes, your program can tell which ones were checked and decide what to do in response.

First, let's make a dialog box with a check box to see how it works. Start with a fresh program, put the cursor in the Main procedure, and start the UserDialog editor. Adding a check box to your dialog box is like adding a button, but instead of using the "Button icon" you use the "Checkbox icon", which looks like a square with an X inside it. Then you can place your check box.

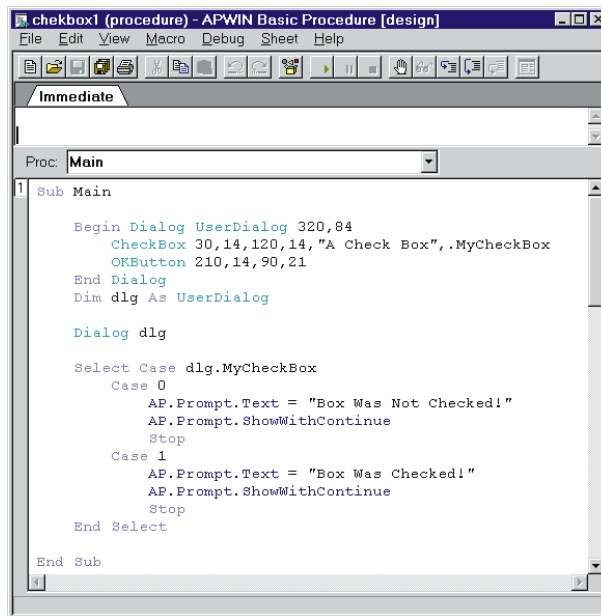
Also, you need to add some sort of button to your menu. This is because a button tells APWIN that you are finished with the dialog box and ready to continue with the program. A check box doesn't - you can leave it as it is, or check and uncheck it as many times as you like before telling the program to continue. If you don't have any other buttons, you need to add an 'OK' button, which just tells the program that you are happy with what you have checked and ready to continue. The "Ok Button icon" is in the bottom left hand corner of the icons. If your program tries to put up a dialog box without any buttons at all, it will give you an error message.

While you are in the UserDialog Editor, double click on the check box. You will get the usual window of details about the check box. Take a look at the item marked "Field" (it should say something like CheckBox1). This is the name your program will use to identify this check box. Change this name to "MyCheckBox". This is sort of like the button number - you need to know what it is in order for your program to use the information. Notice that this can be different from the text shown beside the check box - the displayed text is called the "Caption".

Modify your dialog box so that it looks like this:



Then go back to your program and change it so that it looks like this:

The image shows a window titled "checkbox1 (procedure) - APWIN Basic Procedure [design]". The title bar includes standard window controls and a menu bar with "File", "Edit", "View", "Macro", "Debug", "Sheet", and "Help". Below the menu bar is a toolbar. The main area of the window shows the code for the "Main" procedure. The code is as follows:

```
Sub Main
    Begin Dialog UserDialog 320,84
        CheckBox 30,14,120,14,"A Check Box",.MyCheckBox
        OKButton 210,14,90,21
    End Dialog
    Dim dlg As UserDialog

    Dialog dlg

    Select Case dlg.MyCheckBox
        Case 0
            AP.Prompt.Text = "Box Was Not Checked!"
            AP.Prompt.ShowWithContinue
            Stop
        Case 1
            AP.Prompt.Text = "Box Was Checked!"
            AP.Prompt.ShowWithContinue
            Stop
    End Select
End Sub
```

CHEKBOX1.APB Program

Then save and run your program. It should put up a dialog with a check box in it, and you can check or uncheck it and press Ok. Your program will then give you a message, telling you whether or not the box was checked.

How did we do that?

You have seen most of the techniques before, but they are used in slightly different ways. We define our dialog box (or APWIN defines our dialog box) and then use the “Dialog dlg” command to “make it happen”. When we use a dialog box with check boxes in it, APWIN creates variables (remember them?) to tell us whether our boxes are checked. These variables have a name that is made of the name of the dialog box, then a period, and then the name of the check box. We haven’t worried about the name of our dialog box so far, but in this case the name is ‘dlg’, which is the default name APWIN gives all your dialog boxes. So, we are automatically given a variable called `dlg.MyCheckBox`, which contains a 1 if the box is checked or a 0 if it is not.

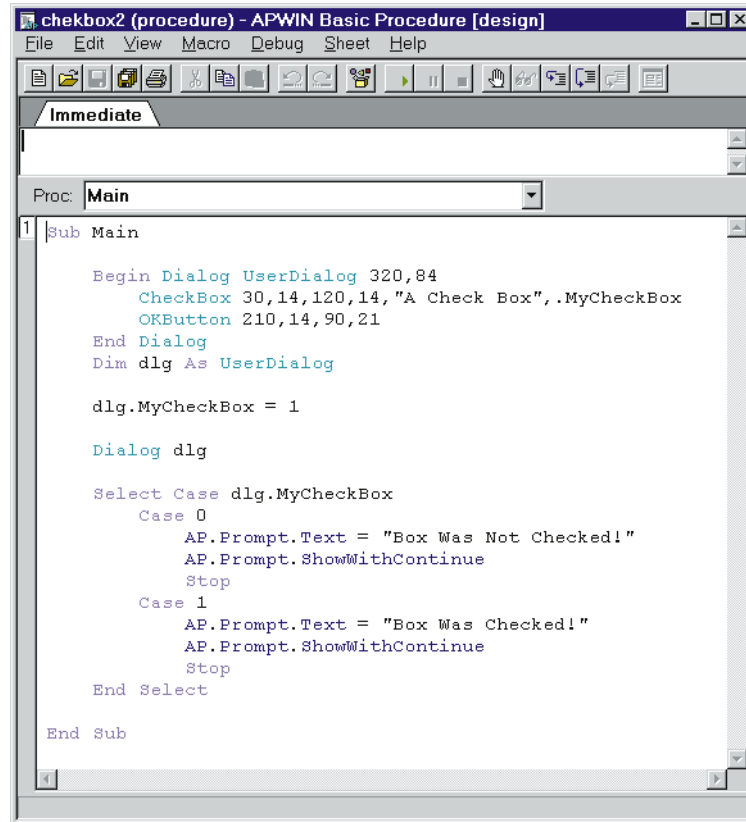
The program then uses this variable in a `Select...Case` statement to decide which message to put up.

Changing your Check Box's Default

As you probably noticed, the check box starts off unchecked. Each time you run the program, no matter how it was when you left it, the box will be unchecked again. This is called the *default state* of the check box, and you can change it to start off checked.

To do this, just set the variable to 1 in the same way that you would set any other variable to 1. You need to do this before starting the dialog box (the “Dialog dlg” statement). The following page shows the same program, but modified so that the check box has a default state of checked.

This program should always start with the check box checked. If you are running a looping menu program (as described previously), you have to be careful about where you put the “`dlg.MyCheckBox = 1`” instruction. This instruction tells it to change the state of the box to be



```
chekbox2 (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help
Immediate
Proc: Main
1 Sub Main
    Begin Dialog UserDialog 320,84
        CheckBox 30,14,120,14,"A Check Box",.MyCheckBox
        OKButton 210,14,90,21
    End Dialog
    Dim dlg As UserDialog

    dlg.MyCheckBox = 1

    Dialog dlg

    Select Case dlg.MyCheckBox
        Case 0
            AP.Prompt.Text = "Box Was Not Checked!"
            AP.Prompt.ShowWithContinue
            Stop
        Case 1
            AP.Prompt.Text = "Box Was Checked!"
            AP.Prompt.ShowWithContinue
            Stop
    End Select

End Sub
```

CHEKBOX2.APB Program

checked. If you put this instruction before the loop, it will only happen once, at the beginning. If you put it inside the loop, it will happen once each loop. If you put it between the “Dialog dlg” command and the Select...Case statement, the Select...Case will always select the state that you just set the variable, which means your program won’t work properly. It is a good idea to think through your program step-by-step and make sure everything happens in the right order.

Expanding the Select...Case Idea

The Select...Case is a very powerful tool, but not always exactly the right tool for the job. Now we will be introduced to a slightly more flexible tool called If...Then.

The If...Then statement is similar to Select...Case, in that it allows your program to decide whether to run certain instructions. The general format is like this:

```
If <expression> Then <instruction>
```

The <expression> will be a mathematical equation, and the <instruction> will only be used if the equation is true. For example, if you had a variable called x, you could use the following:

```
If x = 5 Then MyProcedure
```

The above instruction would look at the variable x, and if it was equal to 5, it would run MyProcedure. If x was anything else, then nothing would be done. The next instruction to be read would be the one right after the If...Then statement.

You can use these instructions for all sorts of things when you are working with dialog boxes. For example, if you have a check box called ExtraTests, and you want it to run some extra tests if the box was checked, you can use the following statement:

```
If dlg.ExtraTests = 1 Then MoreTests
```

Assuming you have defined a procedure called MoreTests that runs the extra tests, this would run the extra tests only if the box was checked.

More on the If...Then

If you want to run more than one instruction based on your If...Then, you use a slightly different format, like this:

```
If <expression> Then
  <instruction>
  <instruction>
  <instruction>
End If
```

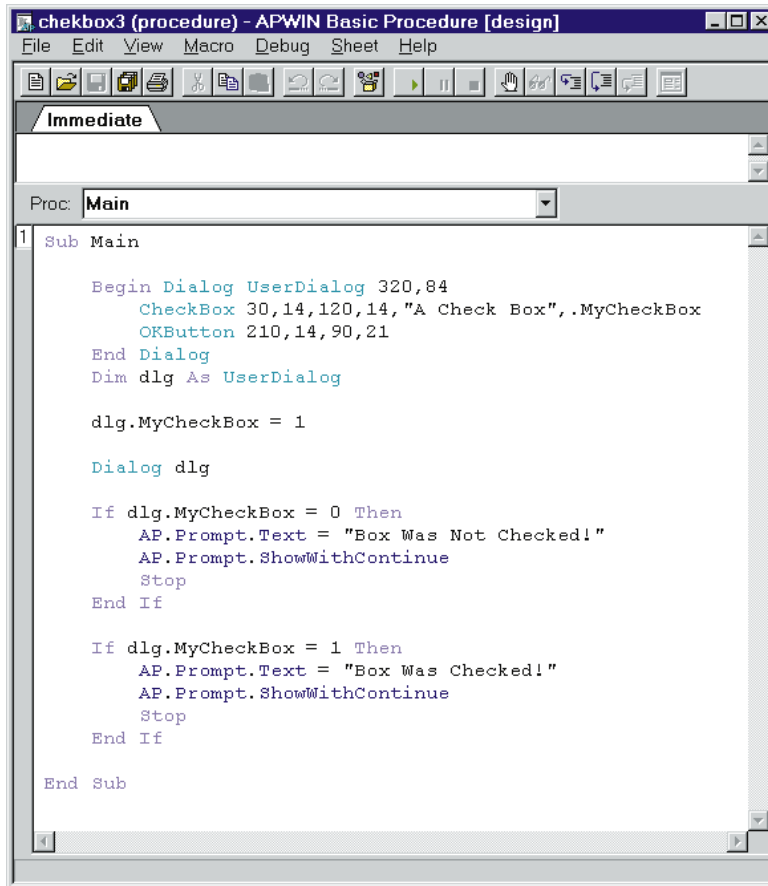
As usual, you will have to fill in the italicized parts yourself, telling your program what to look for and what to do. You can have as many instructions as you want. If the expression is true, then all the instructions down to the End If will be used. If the expression is false, then APWIN will skip down to the End If and start from the next instruction after that.

There is another format you can use, for even greater flexibility. This format will run a certain piece of program if the expression is true, and another piece of program if the expression is false:

```
If <expression> Then
  <instruction>
  <instruction>
Else
  <instruction>
  <instruction>
End If
```

In this case, the first block of instructions, between the words Then and Else, will be used if the expression is true. The second block of instructions, between the words Else and End If, will be used if the expression is false.

Here is the same program used in the previous example, but using two If...Then statements instead of the Select...Case statement:



```
checkbox3 (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help

Immediate

Proc: Main
1 Sub Main

  Begin Dialog UserDialog 320,84
    CheckBox 30,14,120,14,"A Check Box",.MyCheckBox
    OKButton 210,14,90,21
  End Dialog
  Dim dlg As UserDialog

  dlg.MyCheckBox = 1

  Dialog dlg

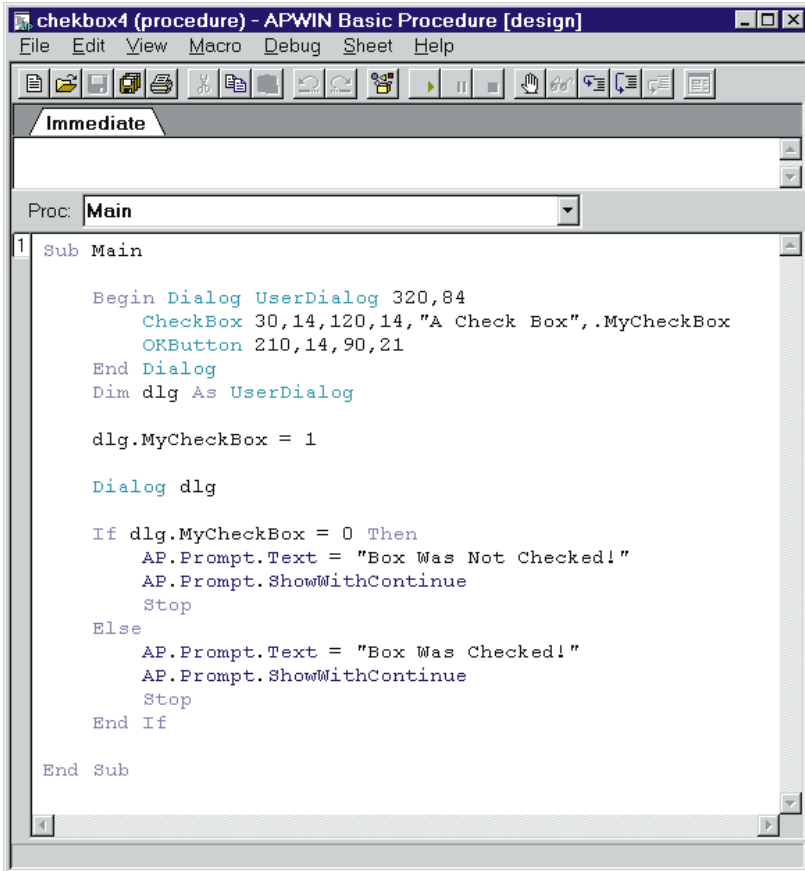
  If dlg.MyCheckBox = 0 Then
    AP.Prompt.Text = "Box Was Not Checked!"
    AP.Prompt.ShowWithContinue
    Stop
  End If

  If dlg.MyCheckBox = 1 Then
    AP.Prompt.Text = "Box Was Checked!"
    AP.Prompt.ShowWithContinue
    Stop
  End If

End Sub
```

CHEKBOX3.APB Program

Here is the same program using If...Then...Else.



```
checkbox4 (procedure) - APWIN Basic Procedure [design]
File Edit View Macro Debug Sheet Help

Immediate

Proc: Main
1 Sub Main
  Begin Dialog UserDialog 320,84
    CheckBox 30,14,120,14,"A Check Box",.MyCheckBox
    OKButton 210,14,90,21
  End Dialog
  Dim dlg As UserDialog

  dlg.MyCheckBox = 1

  Dialog dlg

  If dlg.MyCheckBox = 0 Then
    AP.Prompt.Text = "Box Was Not Checked!"
    AP.Prompt.ShowWithContinue
    Stop
  Else
    AP.Prompt.Text = "Box Was Checked!"
    AP.Prompt.ShowWithContinue
    Stop
  End If

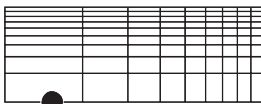
End Sub
```

CHEKBOX4.APB Program

As you can see, there are often several ways to accomplish the same task. The way that you choose depends on what seems comfortable for you, and the job you are trying to do.



Audio
precision



Audio Precision, Inc.
PO Box 2209
Beaverton, Oregon 97075-3070
U.S. Toll Free: 1-800-231-7350
Tel: (503) 627-0832 Fax: (503) 641-8906
email: techsupport@audioprecision.com
Web: www.audioprecision.com



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