

The Fundamentals of Mixed Signal Testing

Soft Test Inc.
246 Union Ave.
Los Gatos, CA 95032
USA
Phone 408.377.1888
Fax 408.377.1866

<http://www.soft-test.com>

Edition 1.08

Copyright ©1998, 1999, 2000 Soft Test Inc. All rights reserved. This document and the associated DSP Lab software are protected by U.S. and international laws and treaties. They may not be copied, reproduced, stored in a retrieval system or transmitted, in any form or by any means, without the prior written consent of Soft Test Inc.

90-0-00049

Introduction

Purpose

What is *mixed signal testing*? In the early days of mixed signal devices, mixed signal testing meant testing an A/D or D/A converter or a track-and-hold device. With the advent of digital signal processing (DSP) techniques for device testing in the 1970s and 1980s, mixed signal testing became a more general way to generate and measure analog signals with digital processing of the resulting data.

The use of DSP techniques with A/D and D/A conversion is the essence of creating fast, repeatable and accurate test programs for mixed signal devices. The purpose of this course is to have you understand:

- ❖ The components of a mixed signal test system
- ❖ The mathematical basis of DSP
- ❖ The principles of analog signal theory
- ❖ The basics of sampling theory and how to correctly sample an analog signal
- ❖ How to create an analog signal for a DUT with a waveform generator
- ❖ How to connect a device under test so that analog and digital signals do not interfere with each other
- ❖ Why and when analog filtering and other conditioning is necessary
- ❖ Typical DSP algorithms and when and how they should be used
- ❖ How to extract test measurements from sampled data and relate them to device specifications

Philosophy

Analog device design and testing has a distinctly different philosophy from its digital counterpart. In digital, the general thought is “If I deliver exactly this input stimulus, I get exactly that output result.” In analog, it is more akin to “If I deliver exactly this input stimulus, I should get an output within this range.” Analog measurements and limit comparisons decide not if a device is exactly right, but if it is *good enough*. Learning to “think analog” will help you be more comfortable with this concept and make it easier for you to comprehend the world of mixed signal testing.

Two devices will be used as examples for DSP testing. They are a D/A converter and an A/D converter. Although there are many mixed signal device types, these two are chosen for several reasons:

1. The output signal from a D/A is an analog signal which could be from *any* analog circuit. Thus the knowledge basis and techniques used to measure its parameters could be used to measure any analog signal.
2. An A/D converter has an analog input, digital inputs and digital outputs. It requires synchronous conversion timing and output data collection plus DSP analysis of output results. It requires input signal filtering and conditioning and may have an input multiplexer or track and hold built in.

3. The equipment training classes given by automatic test equipment manufacturers often use data converters as their example devices. This prepares you to learn more about the equipment by already knowing the fundamental principles of data converters.

A little history

Prior to the 1980s, most integrated circuits contained either analog or digital functions. Circuits which combined, or mixed, analog and digital circuits existed in module and hybrid form and were difficult to manufacture and test and expensive to use. As process and design techniques improved, integrated mixed signal devices proliferated rapidly through all aspects of the integrated circuit manufacturing industry. These *mixed signal* circuits pose a unique set of problems for those responsible for testing them.

Long test times were the norm as slow integrating voltmeters were used for voltage measurements; slow settling analog filters cleaned and purified DUT input signals; averaging techniques were required for repeatability; instruments communicated via slow serial or GPIB protocols, etc. With the cost of fabricating integrated circuits decreasing and their complexity increasing through the 1980s, test cost became a significant portion of their total manufactured cost. It became clear that traditional techniques for performing measurements on the analog portions of a mixed signal device would not have the combined speed, accuracy and repeatability required by the complex mixed signal devices looming on the horizon.

Coincidentally in the 1980s, numerical methods of digitally processing analog information were moving from the laboratory into the main stream of circuit development and testing. These numerical methods, dubbed *digital signal processing* or DSP, offered a fast way of performing the traditional analog measurements, including DC and AC measurement, filtering, amplitude versus time calculations (replacing an oscilloscope), frequency spectra calculations (replacing a spectrum analyzer) and other functions not previously available with traditional instruments. Specialized integrated DSP calculation circuits became available.

The automatic test equipment industry, led by Matthew Mahoney of LTX Corp., began using these techniques for testing analog and mixed signal integrated circuits. DSP based testing replaces many of the traditional analog instruments with 3 processes: waveform generation (D/A conversion), waveform digitizing (A/D conversion), and DSP calculation.

Course Material

This book and its accompanying software laboratory were written for the training course *The Fundamentals of Mixed Signal Testing* taught by Soft Test Inc. It contains text and figures which present a wide variety of real world experience in mixed signal design, manufacture and testing. The math, although substantial, is presented in a way that is not intimidating. Thought provoking questions and lab exercises are interspersed throughout the material. Answers to questions are at the end of each chapter.

To assist in understanding the material, software laboratory experiments are included to demonstrate the concepts of Fourier series, sampling theory, waveform generation, waveform sampling, coherency and time and frequency graphing. The software uses Fast Fourier Transform and Inverse Fourier Transform algorithms and simulates a real oscilloscope and spectrum analyzer. It graphically illustrates the concepts while being fun to use and potentially useful while developing a device test.

The purely digital aspects of mixed signal testing are not covered in detail in this course. If you need to learn about digital vector patterns, timing edge measurement, VOL or VOH tests, etc., please look into Soft Test's course on *The Fundamentals of Digital Semiconductor Testing* or CD-ROM computer based training course on *Semiconductor Testing—Digital Device Fundamentals*. More information is available at our web site—<http://www.soft-test.com>.

DSP Lab software

This course has Microsoft Windows based interactive laboratory exercises which demonstrate the principles of sampling, Fourier series, sinusoidal waveforms, FFT type Fourier transforms, inverse FFT, signal generation and other mixed signal testing concepts. The interactive *DSP Lab* software requires a computer system with the following characteristics:

- ◊ 80486 or higher CPU
- ◊ 24 Meg RAM
- ◊ Windows 95, Windows 98, Windows NT 4 or higher operating system
- ◊ approximately 600K bytes of hard disk space

To schedule a class or for more information contact:

**Soft Test Inc.
246 Union Ave.
Los Gatos, CA 95032**

**408.377.1888 (voice)
408.377.1866 (fax)
<http://www.soft-test.com/>**

**This Document is protected by US and
International Copyright Laws.**

It is illegal to make copies of this material without the written consent of Soft Test Inc. If you would like additional copies please use the Order Form on the following page.

Thank You

Introduction



246 Union Avenue
Los Gatos, CA 95032
Phone: (408) 377-1888
Fax: (408) 377-1866
e-mail: support@soft-test.com

Order Form

Date: _____

Name	
Company	
Address 1	
Address 2	
City, St/Prov	
Country	
Phone	
Fax	
E-Mail	

<input type="checkbox"/> Purchase Order <input type="checkbox"/> Check <input type="checkbox"/> Cash <input type="checkbox"/> Visa or Mastercard		<input type="checkbox"/> Priority mail add \$6.00 <input type="checkbox"/> Add \$3.00 per additional manual for shipping in the U.S.A.
---	--	---

ITEM	DESCRIPTION	PRICE	NOTES
	<i>The Fundamentals of Digital Semiconductor Testing</i> reference manual	\$100.00	
	<i>The Fundamentals of Mixed Signal Testing</i> reference manual with DSP Lab software	\$100.00	
	<i>Digital Device Fundamentals</i> Interactive CBT course	contact us	

Prices are subject to change.

Subtotal

(California only) 8.25% Sales Tax

Shipping

Total Amount Due

Make all checks payable to: **Soft Test Inc.**

Purchase orders can be faxed with this order form to Soft Test at (408) 377-1866.

Visit our web site at www.soft-test.com

THANK YOU FOR YOUR BUSINESS!



Introduction

Table of Contents

Introduction	i
Purpose	i
Philosophy	ii
A little history	ii
Course Material	ii
DSP Lab software	iii
Overview of Mixed Signal Testing	1
Goals	1
Objectives	1
What You Will Learn	2
Digital Signals	3
Two signal levels	3
Information storage with digital signals	3
Other uses for digital signals	5
Digital Circuits	7
Digital Test Systems	9
Functional testing	11
Logic Levels	13
Parametric Testing	14
Vector Sequencing	17
Analog Signals	20
The Time aspect of analog signals	22
Analog Circuits	23
Analog Test Systems	25
Mixed Analog and Digital Signals	29
Mixed Signal Circuits	30
Mixed Signal Test Systems	32
Waveform Digitizer	34
Waveform Generator	35
Waveform Data Storage (Capture) Memory	37
Digital Signal Processor	37
Key Points of This Chapter	39
Answers to Chapter Questions	40
References	42
Basic Analog Specifications	43
Goals	43
Objectives	43
What you will learn	44
Digital Device Specifications	45
Analog Circuit Specifications	46
Op Amp specifications	46
Comparator Specifications	51
Filter Specifications	52
Key Points of This Chapter	54
Answers to chapter questions	55
References	56
Digital to Analog Converter Static Measurements	57
Goals	57
Objectives	57



Table of Contents

What you will learn	58
Overview of Testing DAC Static Parameters	59
DAC Static Specifications	60
Other parameters of interest.....	63
Test System Configuration for DAC Static Parameter Tests	64
Example DAC data sheet	65
Parameter Measurement Requirements	66
Measuring offset and gain errors	67
Zero scale measurement conditions	67
Offset Error.....	69
Gain Error.....	72
Measuring DNL and INL.....	74
Making one DNL measurement	74
Making all DNL measurements.....	75
What is Superposition Error?	78
Faster Measurements.....	79
The DNL Test	82
The INL Test.....	83
Current Output DAC.....	84
Key Points of This Chapter	85
Answers to chapter questions	86
References.....	88
 Analog to Digital Converter Static Measurements	89
Goals	89
Objectives	89
What you will learn	90
Overview of Testing ADC Static Parameters	91
ADC Static Specifications	92
Other Important Parameters	95
Test System Configuration for ADC Static Parameter Tests	97
Example device specification	98
Parameter Measurement Requirements	98
The Digital Side of ADC Testing	100
Unique ADC Testing Problems	102
Transitions, zero scale, full scale and LSB size.....	102
ADC transition points	104
Calculating Offset Error	107
Calculating Gain Error	108
DNL and INL Testing	109
Histogram Test for DNL and INL.....	109
Using an intelligent feedback loop to test DNL and INL	113
Which codes to test?	114
Finding the “center of code”	115
Static DNL.....	116
Static INL.....	117
Effective Number of Bits (ENOB)	118
Key Points of this Chapter	119
Answers to chapter questions	120
References.....	121
 The Mathematics of DSP	123
Goals	123
Objectives	123

Table of Contents



What you will learn	124
Logarithms and exponents	125
Logarithm bases	125
Properties of logarithms.....	127
Conversion between bases	128
Decibels (dB)	129
Time and Frequency	131
Periodic Motion.....	131
Rotating vector	134
Converting angle θ to frequency and time	136
Phase and the cosine function.....	138
RMS value of a sinusoid	140
Time to frequency translation.....	143
Frequency domain plot of DC.....	143
Frequency domain plot of a sine wave.....	144
Frequency domain plot of an impulse.....	145
Fourier series	146
Dirichlet conditions	146
Calculating the sine and cosine components.....	146
Creating and examining a Fourier series.....	148
Complex numbers.....	155
The j operator	155
The complex plane.....	155
Key Points of This Chapter	159
Answers to chapter questions	160
Answers to Lab Exercise 5.1	161
References.....	162
 Sampling.....	163
Goals	163
Objectives	163
What you will learn	164
Requirements for sampling	165
Shannon's theorem	165
Nyquist's limit	165
Periodicity	167
What does a sample set represent?.....	167
Converting a time sample set to frequency.....	168
Discrete Fourier transform (DFT)	169
Fast Fourier transform (FFT)	169
Using an FFT algorithm	170
Problems that can occur with sampling	172
Spectral replication and aliasing.....	172
Leakage.....	175
$\text{Sin}(x) / x$ Amplitude Error.....	179
Truncation error	181
Quantization Error.....	181
Slew Rate Error.....	182
Jitter Error	183
Coherent sampling.....	184
What is coherent sampling?	184
Why sample coherently?.....	184
Coherency relationships	184
Another benefit of coherent sampling	187

Table of Contents

Digitizing samples	189
Sampling a Waveform	189
Frequency analysis of samples	192
Creating a Frequency Spectrum from Digitized Waveform Samples	192
Generating time samples.....	197
Using a software algorithm	197
Using an Inverse Fourier Transform	197
When the IFFT is most useful.....	198
How to use an Inverse FFT (IFFT) algorithm	198
Limitations of using the Inverse Fourier Transform	200
Generating Time Samples with Inverse Fourier Transform	200
Key points of this chapter	203
Answers to chapter questions	204
Answers to Lab Exercise 6.1	205
Answers to Lab Exercise 6.2	206
Answers to Lab Exercise 6.3	207
References.....	208
Digital to Analog Converter Dynamic Parameters	209
Goals	209
Objectives	209
What you will learn	210
DAC Dynamic Specifications	211
Signal to noise and distortion ratio (SNDR or SINAD).....	214
Signal to noise ratio (SNR).....	215
Total harmonic distortion (THD)	216
Intermodulation Distortion (IM)	217
Other parameters needed for dynamic testing.....	220
Testing DAC Dynamic Parameters	221
Test System Configuration for DAC Dynamic Parameter Tests.....	222
Example DAC Data Sheet.....	223
Parameter Measurement Requirements	224
Items and Steps Required to Dynamically Test a DAC	225
Creating the DAC Output Signal	226
Calculating the digital input signal as an array of points.....	226
Creating DAC inputs for a sine wave	231
Filtering the output signal	235
Capturing the DAC output.....	239
Real World Example of Digitizing a Sine Wave.....	245
Calculating the result parameters	247
Overall gain correction.....	247
Time to frequency conversion.....	247
Fundamental correction	248
Creating a Spectrum Graph.....	250
Synchronization Issues	251
Key Points of This Chapter	253
Answers to chapter questions	254
Answers to Lab Exercise 7.1	255
Answers to Lab Exercise 7.2	256
References.....	257
Analog to Digital Converter Dynamic Parameters	259
Goals	259
Objectives	259



What you will learn	260
ADC Dynamic Specifications	261
Testing ADC Dynamic Parameters.....	263
Test System Configuration for ADC Dynamic Parameter Tests	264
Example ADC Data Sheet.....	265
Parameter Measurement Requirements	266
Items and Steps Required to Dynamically Test an ADC.....	267
Creating an input signal.....	268
Capturing the digital output data	272
Acquiring and holding the input signal	273
Sampling with the ADC under test	279
Coherently sampling a sine wave.....	281
Undersampling.....	284
Undersampling with the beat frequency method	288
Undersampling with the Envelope method.....	290
Calculating SINAD, THD, SNR and IM	292
Sine Histogram Testing	293
Effective Number Of Bits (ENOB).....	297
Key Points of This Chapter	302
Answers to chapter questions	303
Answers to Lab Exercise 8.1:	304
Answers to Lab Exercise 8.2:	305
References.....	306
 General Test Issues.....	307
Goals	307
Objectives	307
What you will learn	308
Does the measurement reflect the state of the DUT or the test system?.....	309
System noise	309
DUT Noise	310
Settling time errors	311
The test system must be 10 times better than the DUT.....	311
What is ground?	312
The tester's earth ground connection	314
DUT power supply	316
Power Supply Decoupling	316
Temperature Effects	318
DUT Reference Signal.....	319
Internal Reference	319
External Reference	319
Averaging and repeatability	320
Trouble-shooting: look for obvious problems first	321
Unexpected things can affect measurement results	322
An ungrounded DUT package	322
External noise from RF or magnetic sources.....	322
Light	322
Humidity	323
Answers to chapter questions	324
References.....	325



Table of Contents

Overview of Mixed Signal Testing

Goals

- ❖ Review the basics of digital devices
- ❖ Review the basics of analog devices
- ❖ Review the components in a typical digital test system
- ❖ Review the components in a traditional analog test system
- ❖ Thoroughly explain the components of a mixed signal test system

Objectives

- ❖ Provide a baseline knowledge of digital, analog and mixed signal test systems as required for the remainder of the course
- ❖ Understand the common elements among digital, analog and mixed signal test systems
- ❖ Understand the distinctions between a mixed signal ATE system and traditional digital and analog ATE
- ❖ Understand the benefits offered by a true mixed signal test system when compared to traditional methods of measuring and testing analog signals

What You Will Learn

- ❖ Logic levels and their corresponding voltage levels (for CMOS/TTL)
- ❖ What is a digital vector (inputs, outputs and I/O)
- ❖ Representing a number as a digital word
- ❖ Digital data as static information or time encoded information
- ❖ Serial and parallel data formats
- ❖ Digital device examples
- ❖ Digital ATE systems and their components
- ❖ PMU operation
- ❖ Binary search algorithm
- ❖ Functional testing
- ❖ Analog signals with infinite levels and time encoded information
- ❖ Basic analog circuits
- ❖ Traditional analog ATE architecture
- ❖ Mixed signal device examples
- ❖ Mixed signal ATE systems and their components
- ❖ Primary difference in a traditional analog test system and a mixed signal test system
- ❖ Purpose of a Waveform Digitizer
- ❖ Purpose of a Waveform Generator
- ❖ Purpose of a Digital Signal Processor

NOTES:



Digital Signals

Two signal levels

Digital devices are easy to classify. They have signals with 2 levels; one represents an *ON* state and one represents an *OFF* state. The origin of the word *digital* is from Latin *digitus*, a finger or toe, so “digital” could mean 10 states, as the states represented by counting on our fingers, rather than just 2. However, our usage here concerns *binary* digital logic so digital circuits and signals we discuss will have 2 levels. Changing a level means going from a logic 1 to a logic 0 or *vice versa*; there is not any other valid information state and no “in-between” level which represents information. (The astute thinkers are now considering a *3 state* signal which has a high impedance state. Does a high impedance state represent information?)

A single state represented by a logic 0 or logic 1 is a “*binary digit*,” which has been condensed to “bit.” A group of 8 bits is a “*byte*” and, generally, 2 bytes is a “*word*” and 2 words a “*double word*.” Sometimes 4 bits is called a “*nybble*.” Different computer languages use different ways to describe register lengths, so the definition of these terms varies.

Information storage with digital signals

Digital data is used to represent information; a group of logic levels contains information, whether the levels are stored on a magnetic disk, in a DRAM register or as holes in a punched paper tape. The information is valid no matter how much or how little time passes, as opposed to analog signals that must change over time to encode and decode information.

When digital signals do change with time, they have an abrupt, instantaneous change from one state (signal level) to another. Ideally, there is no signal between logic 0 and 1. Real circuits have a finite rise and fall time so there is a high speed ramp signal in the transition from one logic state to another. (In poorly designed circuits, the signal can overshoot, ring or move very gradually to the other level as seen in Figure 1.1.)

Serial and parallel data

Digital information is encoded primarily as sequences or groups of levels. A sequence of levels versus time is information encoded as *serial* data while a set of levels in a register is *parallel* data. This data can be encoded in many schemes, e.g. as a one’s or two’s complement number, as an ASCII character or as a set of control bits.

How is a sequence of data clocked from a set of parallel registers classified? This type of digital data can represent almost anything which can be quantified, e.g. a state machine, a computation or a digital representation of an analog signal.

NOTES:

Digital Signals

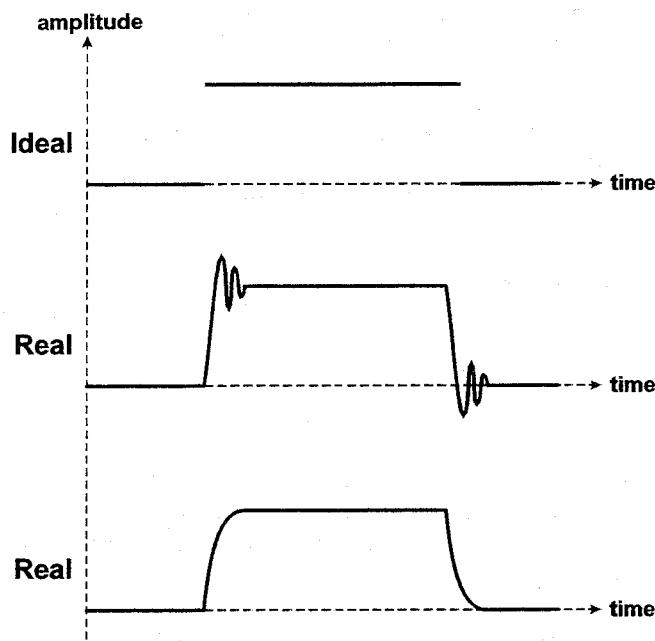


Figure 1.1 Ideal and real digital signals

Using digital data this way makes it simple to encode, decode and store information. Doing complex arithmetic operations is a matter of creating the right set of logic for basic binary mathematical operations and the right set of algorithms for entering data and storing and processing preliminary results into a final answer.

Clocked serial or parallel data implies information that changes with time. Next comes a corresponding consideration of speed, or how much data can be delivered, processed or sent in a given amount of time. Information speed currently tops out at a data frequency of a few hundred MHz for purely digital data. Digital data represented as analog information can be sent and received at GHz speeds and occurs in applications such as digital cellular telephones.

NOTES:



Other uses for digital signals

Control signals

In addition to containing information, digital signals are also great for controlling information flow and physical devices. This is one aspect of digital data that makes its combination with analog signals so powerful.

Number storage

A set of ones and zeros in a register can represent the expected inputs and outputs of a DUT (pure digital data); they can represent the states of a group of relays (control signals); and they can represent a number. The notion of storing numbers as binary digital data is very important in DSP based testing, as the numbers represent amplitude values of an analog signal at a particular time.

As numbers, a binary value can be specified to represent a floating point decimal number, an integer decimal number or other number formats. Floating point numbers are usually specified with a *mantissa* (the fractional part) and an *exponent* (usually an exponent of 10). The binary point is specified to be at the left end of the mantissa, with the mantissa adjusted so that its MSB is in the first location and the exponent adjusted appropriately. The table below shows a binary register and its value as a decimal integer and a decimal floating point value (in IEEE P754 floating point format with 8 bit exponent and 24 bit fraction).

Binary	Unsigned Integer	IEEE P754 Floating Point
101101011010110100111001100110	3048033894	-1.2368383x10 ⁻⁶

Integers can be stored as sign-magnitude, two's complement, one's complement or unsigned values. They can have varying range, depending on the register length used to store them; the longer the register the larger the range. The form of integer (e.g. two's complement) is determined by the way the host computer does its mathematical operations such as addition and subtraction.

Question 1.1: What are the primary characteristics of a digital signal?

NOTES:

Question 1.2: One binary digit is called a

Question 1.3: What are some uses for binary signals?

Question 1.4: What are two ways of representing numbers?

af

NOTES:



Digital Circuits

The *logic* portion of binary digital logic is a phrase describing the way information is represented with digital signals. The most basic form of this logic was created by British mathematician George Boole and is called Boolean logic. Boolean logic lets us represent states of a digital signal with specifically defined relationships and gives us analysis and design tools to create a set of digital data which represents specific information.

A single quanta of information is represented as a logic 0 or a logic 1. Basic devices which process logic 0 and 1 states are the familiar logic gates such as AND, OR, INVERT, NAND, NOR, XOR et al. Each has a transfer characteristic (a.k.a. logic table) which describes a unique output corresponding to each unique input combination. Any logic system can be built with combinations of only NAND or NOR gates, although other basic blocks make logic diagrams easier to read and device logic more compact and efficient.

The basic digital building blocks shown in Figure 1.2 can be combined to create any digital system, from a simple decoder like a binary to 7 segment display circuit to a complex state machine like a microprocessor. The D flip flop shown is a basic memory element; its *Q* output value, once set, does not change until a *Clk* clock signal occurs. The condition of a group of dependent storage elements is called their *state*; the state changes as the clock signal occurs. A system with a set of states is known as a *state machine*. Each bit in a register of a state machine has two possible states, and a state machine with n bits has 2^n possible logic states. A state machine requires a clock and the next state usually depends on the current state and possibly an external stimulus or a previous state. In a test system, states can be represented by a sequence of groups of ones and zeros called the *truth table* or the *test vectors*.

Question 1.5: How many states can a state machine with 16 bits have?

NOTES:

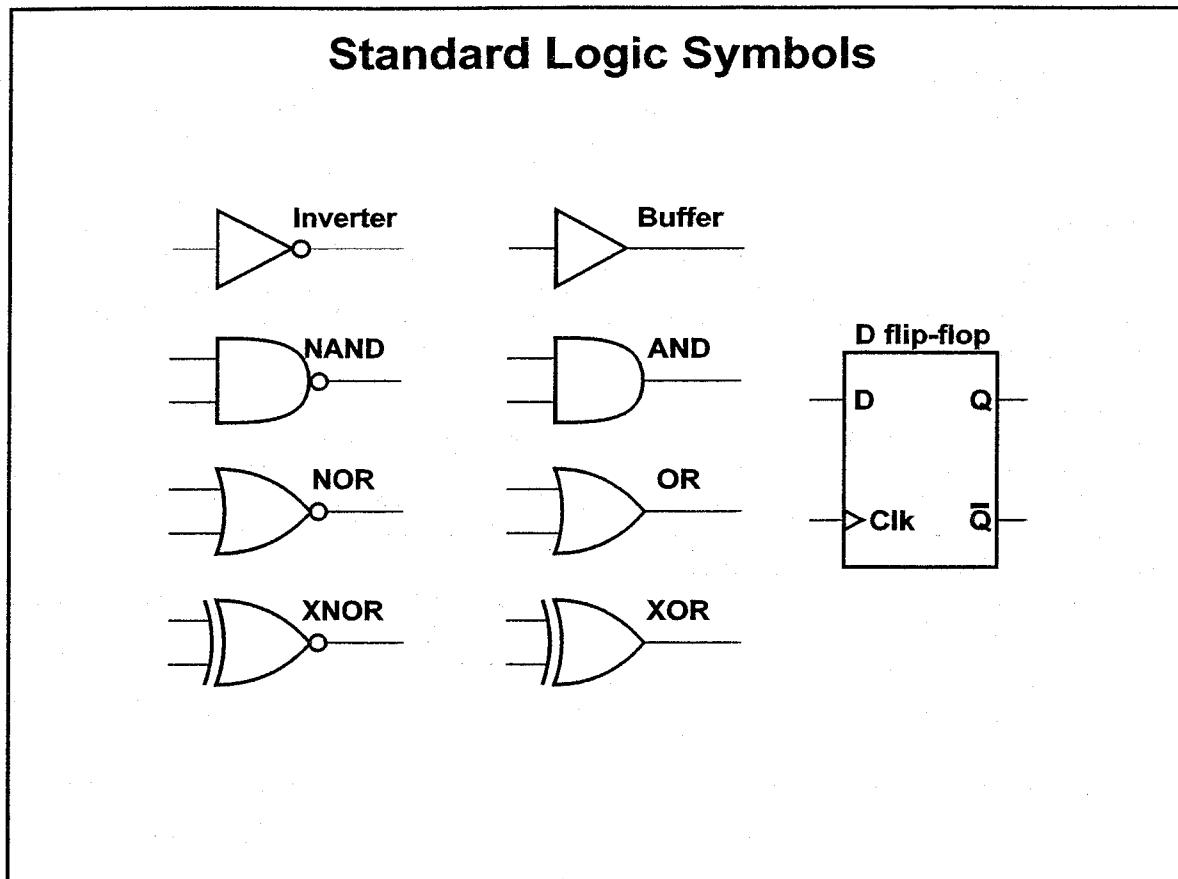


Figure 1.2

NOTES:

Digital Test Systems

Digital test systems were the first truly automated high speed device testers. They were created initially for testing such now-mundane devices as quad AND gates or dual D flip flops. Modern complex devices such as microprocessors and memory still require the same basic operations in a digital test system, to evaluate device functionality and device conformance to DC and AC specifications. Figure 1.3¹ illustrates the basic digital test system components.

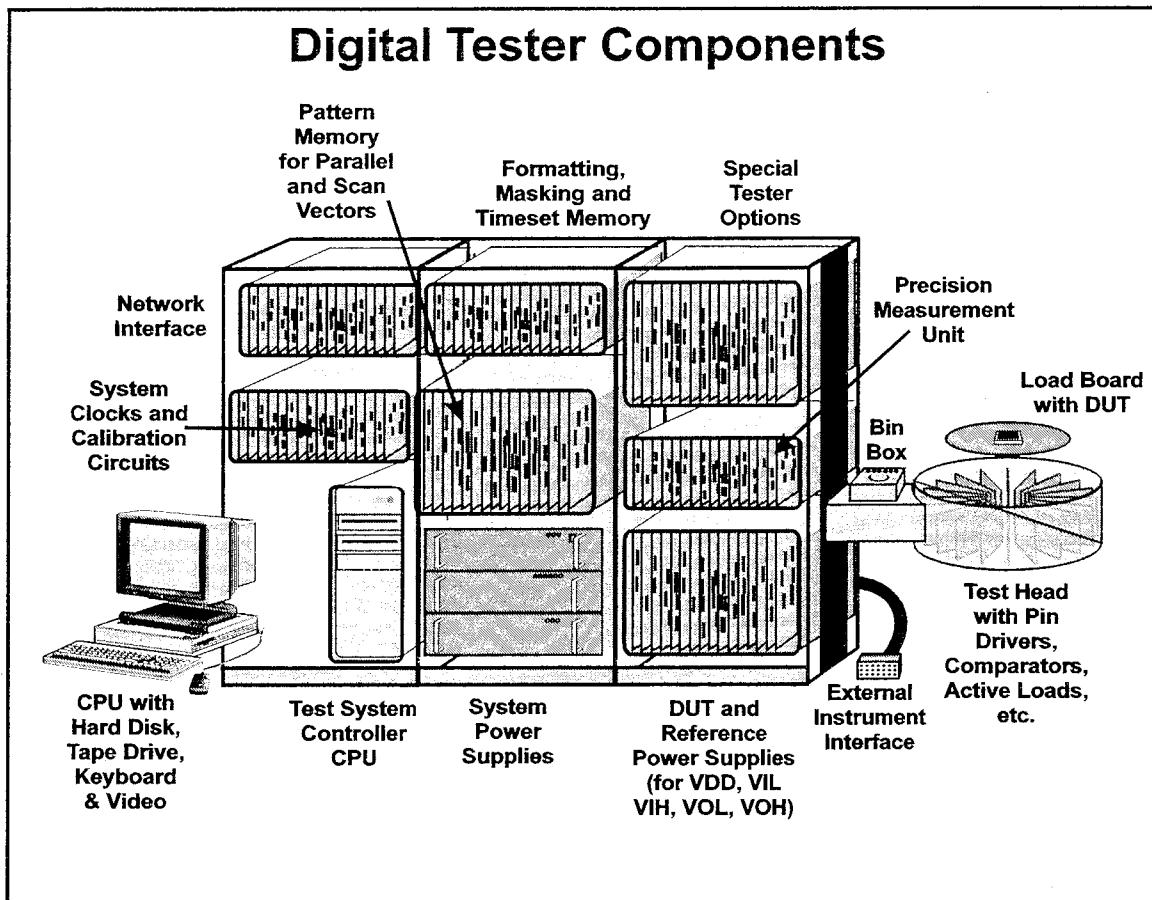


Figure 1.3

NOTES:

Overview of Mixed Signal Testing

The sections shown in Figure 1.3 work together to provide the input stimuli and measure the output results of the digital pins of a device as shown in Figure 1.4.

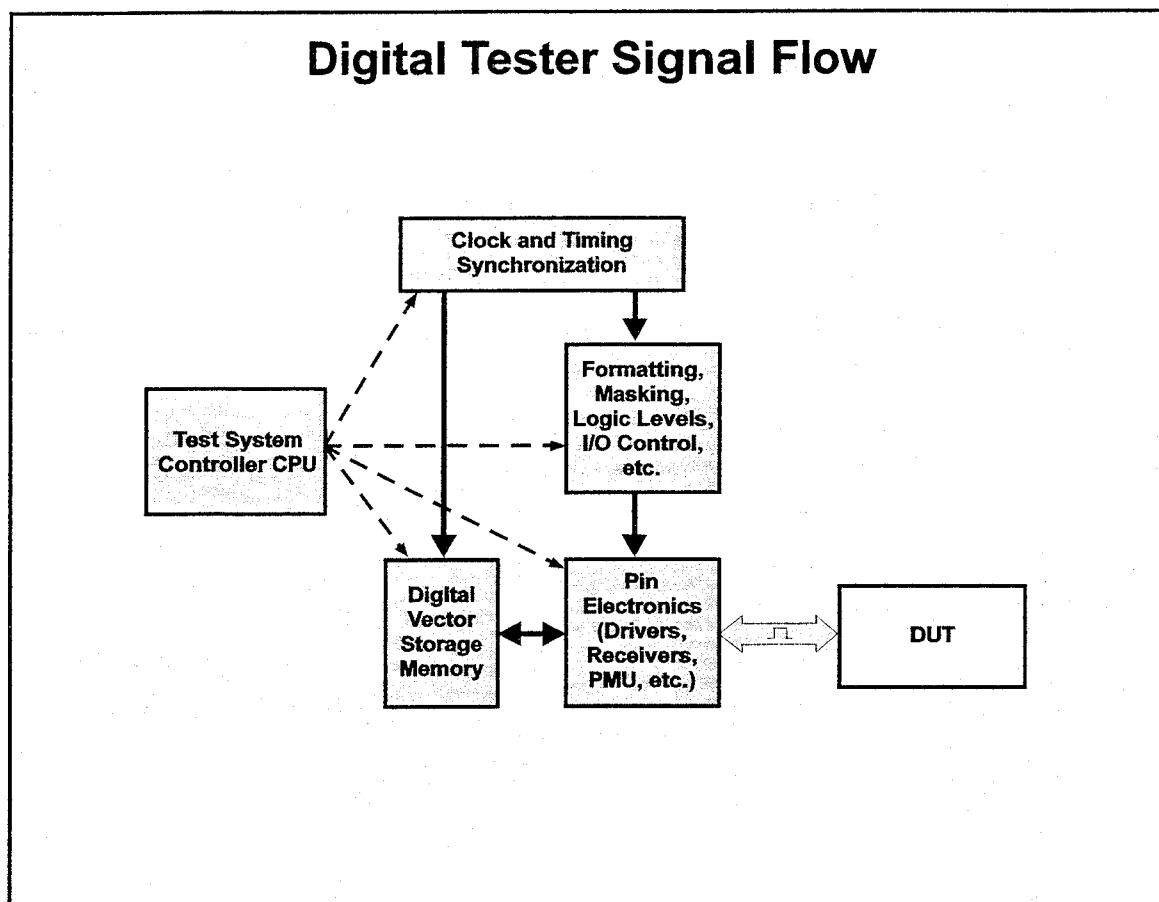


Figure 1.4

Notice that the Test System Controller CPU controls all the various tester components. The stored digital vector data is conditioned by the pin electronics using conditions set by the "Formatting, Masking, Logic Levels, I/O Control etc." block. The DUT interacts with the pin electronics, receiving time, current and voltage formatted signals from drivers and sending its output signals to comparators and/or the PMU.

NOTES:

Functional testing

Evaluation of device functionality (functional testing) verifies that a device's transfer function (its output-to-input relationship(s)) matches the one for which it was designed. In other words, for all known possible input conditions, do the outputs match what is expected? This type of test is generally done as a *Go-No go* test, in which a device test is cancelled as soon as it has a single mismatch between what is expected and what is output. This type of test often uses a parallel set of digital bits called a *vector* to drive device inputs and compare device outputs to what was expected. This requires *vector memory* to hold the digital vector pattern(s). A quad 2-input AND gate requires at least 4 vectors which are 12 bits wide:

4 vectors: 1 vector for each of 4 input logic states

12 bits wide: (2 inputs + 1 output) times 4 gates

A set of bits presented at one time to a DUT is often called a *parallel test vector* and a group of vectors is called a *vector pattern*. A modern microprocessor may require millions of vectors that are tens or hundreds of bits wide, existing in many vector patterns. These vectors are sent to a device at as close to the device maximum speed as possible. When a digital device is tested functionally, a set of vectors is presented to the inputs of the device at maximum data frequency while monitoring the outputs for the expected high and low levels. For a quad 2 input AND gate, the vector table could be:

Pin Type (Input or Output)	I	I	O	I	I	O	I	I	O	I	I	O
Vector 1	0	0	0	0	1	0	1	0	0	1	1	1
Vector 2	0	1	0	1	0	0	1	1	1	0	0	0
Vector 3	1	0	0	1	1	1	0	0	0	0	1	0
Vector 4	1	1	1	0	0	0	0	1	0	1	0	0

NOTES:

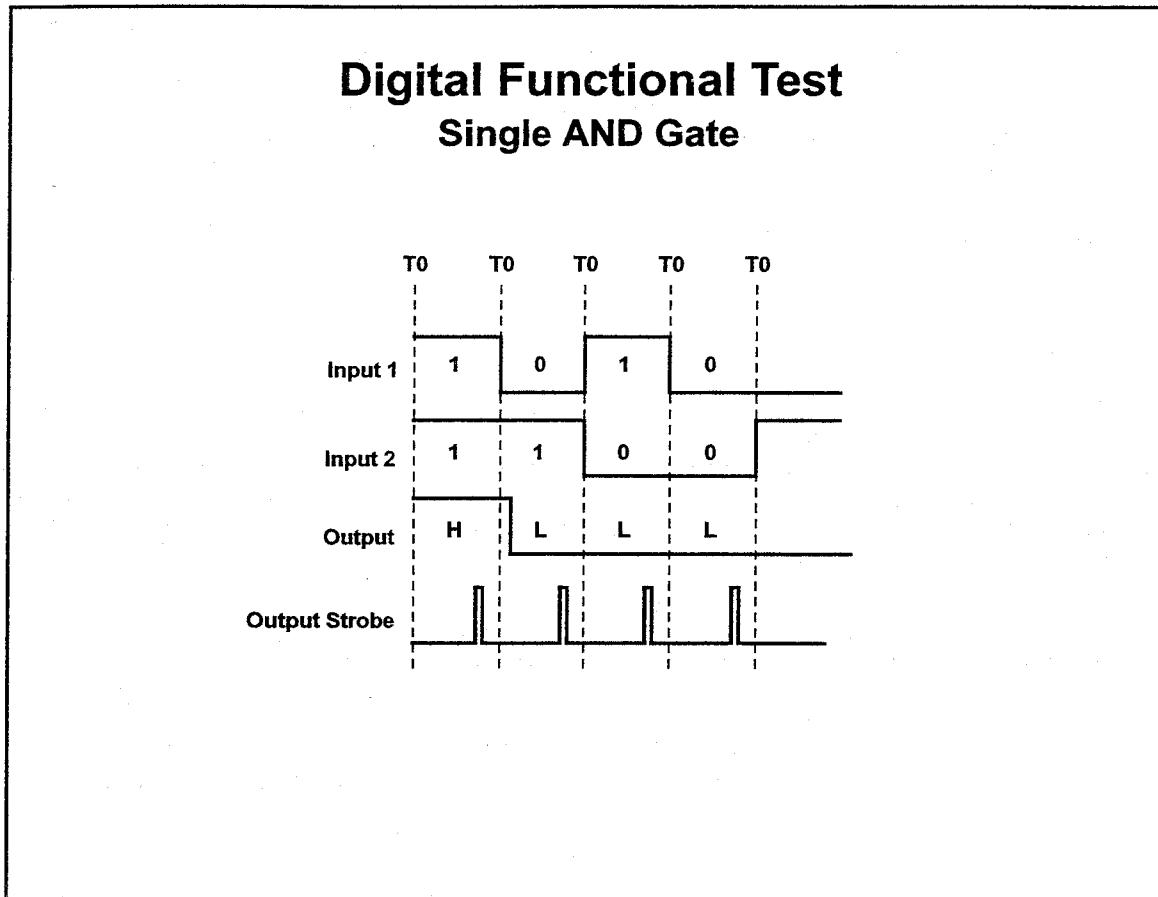


Figure 1.5

Each vector contains 12 logic states, 8 input states and 4 output states. Because each dual input AND has only 4 possible internal states, only 4 vectors are required to test all pins. It would be handy to add additional vectors which condition the inputs and outputs to logic 0 and logic 1 as necessary for DC parametric tests. This allows multiple pins to be tested at a single vector location, simplifying test software development.

NOTES:

Logic Levels

The logic levels of inputs and outputs are verified during functional testing by placing input levels at worst case conditions and monitoring output levels to make certain that they are within specification limits. The only information regarding output levels is whether they are above a minimum output high voltage or below a maximum output low voltage; no quantitative values are known. For example, you will know that a good device always had an output low voltage below 0.4 volts, but you will not know if it was 0.02V or 0.37V.

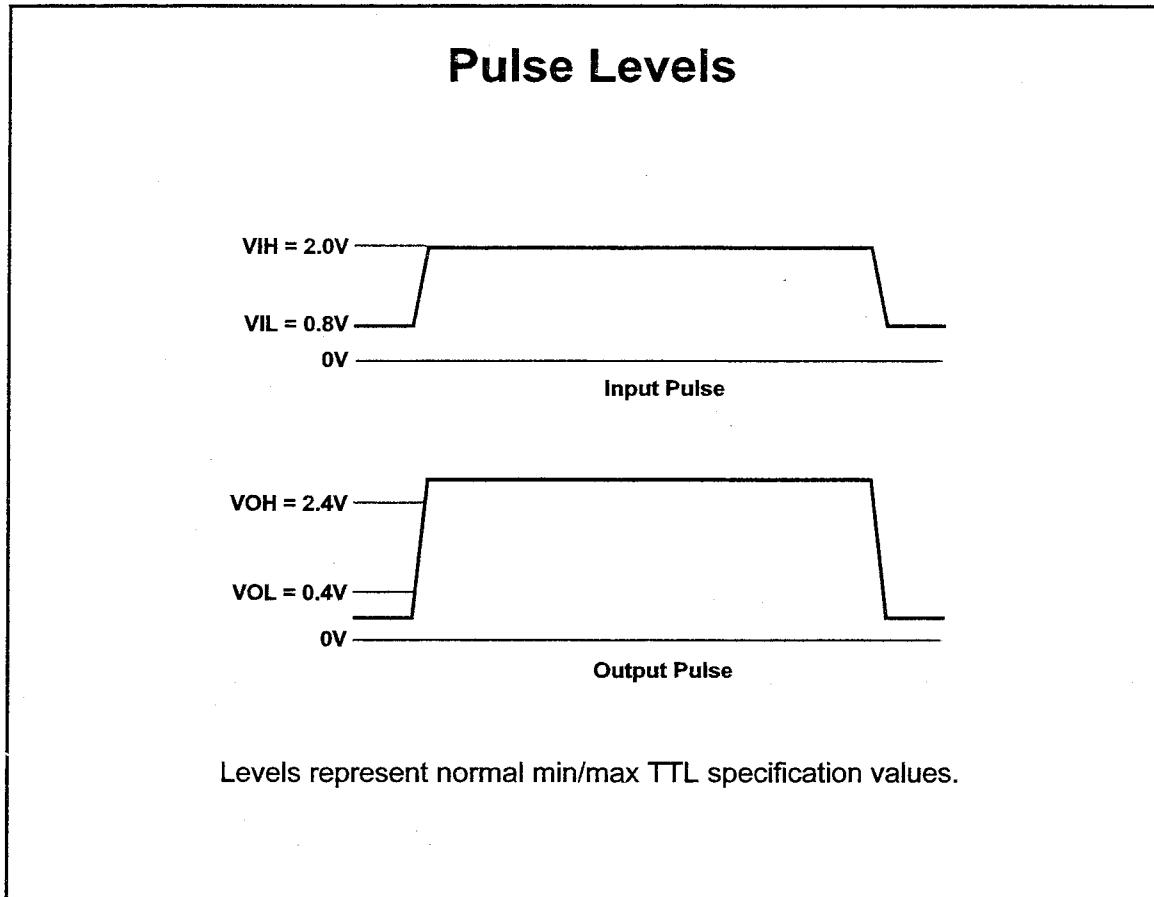


Figure 1.6—Digital input and output signal voltage levels

NOTES:

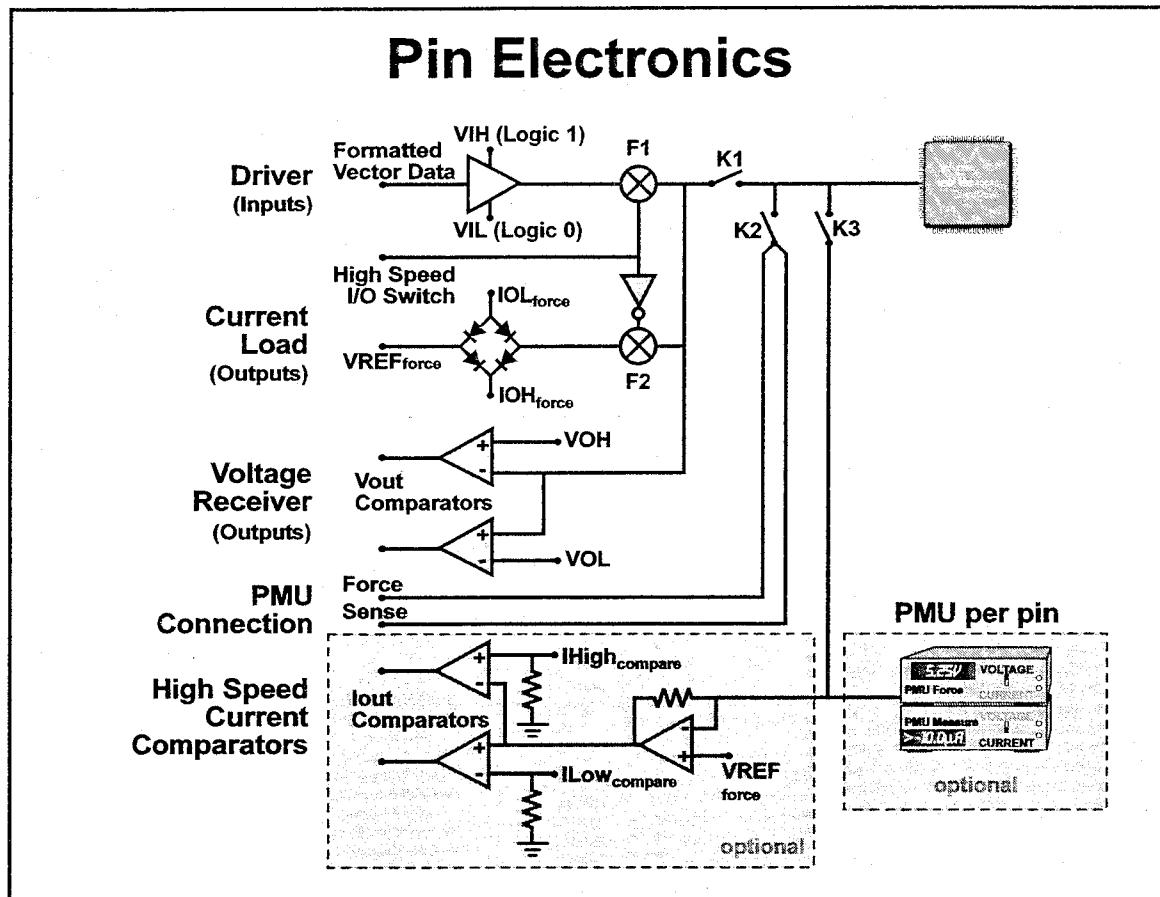


Figure 1.7

Functional input levels are generated by *pin drivers* which condition a device input signal based on the vector state (low or high) and the required input level (VIL for low or VIH for high). Different applications and different device technologies require different input voltage levels to represent logic 0 and logic 1. Input signal conditioning and output signal comparison are done by a *pin electronics card* (PE card) as shown in Figure 1.7.

Parametric Testing

To get precise data on a pin's condition, *parametric* testing is done. Device conformance to DC and AC specifications is known as parametric testing because device *parameters* are measured and compared against their

NOTES:

corresponding specification parameters. Examples of parameters are voltage out low (VOL), input leakage current (IIL), power supply current (IDD), et al.

DC Measurements

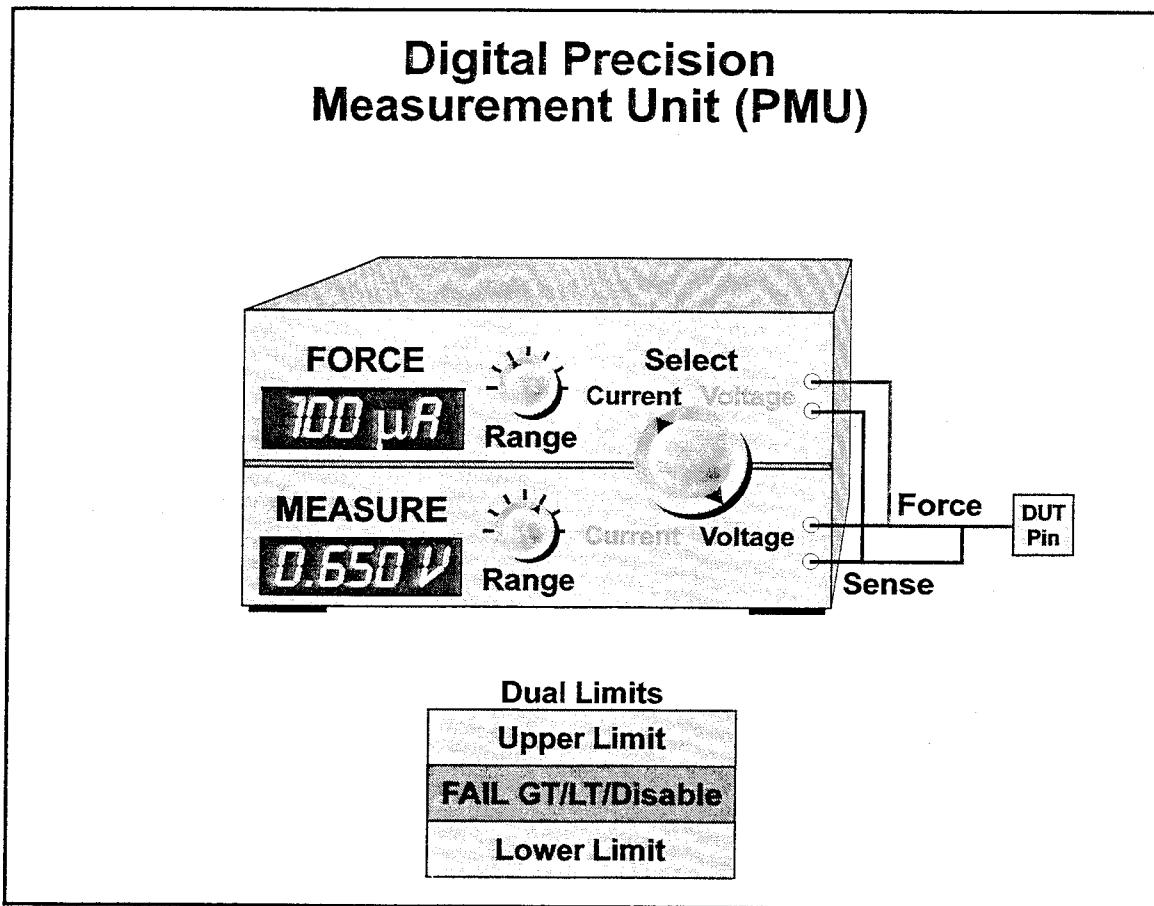


Figure 1.8

A special instrument called a *parametric measurement unit* (PMU), (sometimes also called a *precision measurement unit*) measures device DC parameters in a digital test system. It makes measurements by forcing a voltage and measuring the resultant current or forcing current and measuring the resultant voltage. In essence it treats a given device node as a resistor and makes a measurement of resistance using Ohm's law of voltage = current

NOTES:

times resistance. Often, a device must be *preconditioned* with a functional pattern to put pins in a specific state (high, low or high impedance) prior to making a DC parametric test.

AC Measurements

AC (timing) measurements can be *absolute* or *relative*. An absolute time is measured by starting a timer with a trigger signal and stopping the timer with a second trigger signal. An example of an absolute time measurement is to start a timer with the rising edge of a pulse and stop it with the falling edge. The timer holds the pulse width. The timer must be able to measure time increments which are much less than the time being measured, just like a stop watch is necessary to time a 100 meter dash rather than a wall clock. This can make absolute time measurement difficult for very fast signals. Not all test systems have an absolute time measurement unit.

Relative time measurements can be done with standard functional test equipment. This technique uses a functional pattern to produce an edge which occurs within a test period. The pass/fail strobe is moved until the signal is seen to change state (from a pass to a fail or vice-versa). The difference between the beginning of the test period and the strobe time at the change of state is the measured time value. Setup and hold times are examples of relative time measurement. The resolution of a relative time measurement is only as good as the minimum time increment of the pass/fail strobe.

A fast technique to find the time location of a signal edge within a period is the *binary search* as illustrated in Figure 1.9. This technique requires multiple passes through a single vector. It looks for a pass at the start of the test period then a fail at the end. It then divides the time between output strobes by 2 and subtracts that from the previous strobe time for a failure or adds it for a pass. This process continues until the time to be added or subtracted is less than the resolution of the pass/fail strobe. The time at which the edge occurs is thus known to within the comparator strobe timing resolution.

NOTES:

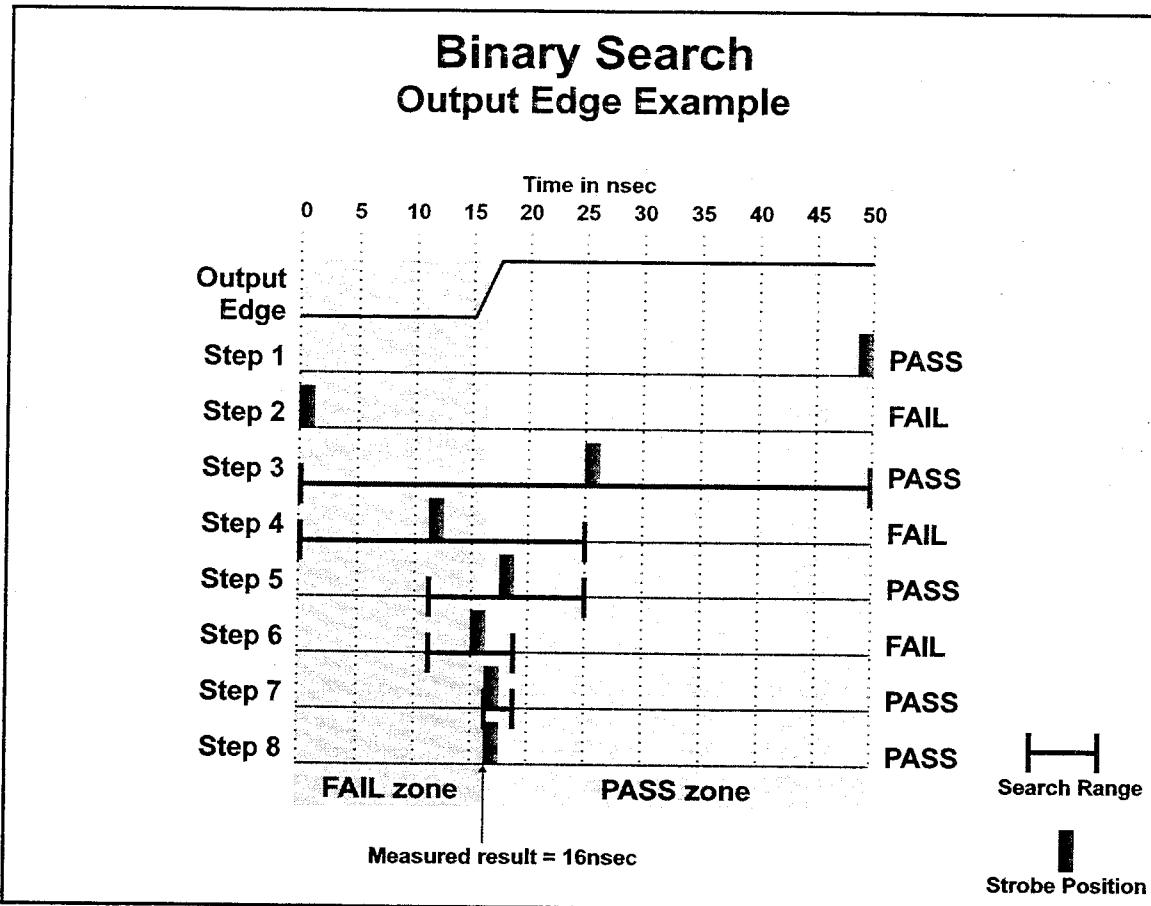


Figure 1.9

Vector Sequencing

The vectors for an AND gate as discussed on page 14 can be executed in a 1,2,3,4 sequence by storing the vectors in 4 sequential memory locations and incrementing through those locations. Some digital test systems have *vector sequencing memory* which allows random sequencing through stored vectors. This sequencing memory can be considered a set of *pointers* to vector memory. To run the vectors in a 1,2,3,4 sequence as before, store the memory addresses in that order in the sequencer memory. Or run the vectors in any order you choose; sequencer memory often has the ability to loop on a vector or set of vectors for a given count and may

NOTES:

Overview of Mixed Signal Testing

have other features such as "loop break on trigger" which cancels a looping vector when a certain pin goes low or high, etc.

Digital Sequencer Memory																														
Direct Memory Addressing		Sequenced Addressing																												
<table border="1"><thead><tr><th>Seq</th><th>Vectors</th></tr></thead><tbody><tr><td>1</td><td>000010100111</td></tr><tr><td>2</td><td>010100111000</td></tr><tr><td>3</td><td>100111000010</td></tr><tr><td>4</td><td>111000010100</td></tr></tbody></table>		Seq	Vectors	1	000010100111	2	010100111000	3	100111000010	4	111000010100	<table border="1"><thead><tr><th>Seq</th><th>Addr</th><th>Vectors</th></tr></thead><tbody><tr><td>3</td><td>1</td><td>000010100111</td></tr><tr><td>4</td><td>2</td><td>010100111000</td></tr><tr><td>2</td><td>3</td><td>100111000010</td></tr><tr><td>2</td><td>4</td><td>111000010100</td></tr><tr><td>1</td><td></td><td></td></tr></tbody></table>	Seq	Addr	Vectors	3	1	000010100111	4	2	010100111000	2	3	100111000010	2	4	111000010100	1		
Seq	Vectors																													
1	000010100111																													
2	010100111000																													
3	100111000010																													
4	111000010100																													
Seq	Addr	Vectors																												
3	1	000010100111																												
4	2	010100111000																												
2	3	100111000010																												
2	4	111000010100																												
1																														

Figure 1.10

Most test vectors today are automatically generated by device simulation software. Simulation software does not generate test vectors with sequencing information thus sequencer memory is often underutilized. It is quite useful in a mixed signal test system to create complex analog waveforms or repeating wave sections. For example, a set of vectors which are the amplitude values for a single sine wave cycle can be looped and sent to a waveform generator to create a sine wave input for a DUT.

Question 1.6: VIHmin = 2.0V and VOHmin = 2.4V for a TTL signal. What does the difference of 0.4V represent?

- a. VOL
- b. Noise margin
- c. Error voltage

Question 1.7: Why are these values used for testing VIH and VOH on a device rather than VIH = 5V and VOH = 4V?

NOTES:

Question 1.8: What are the 3 major types of tests performed on digital device pins?

Question 1.9: Of the major test types from Question 1.8, which use the PMU?

Question 1.10: Of the major test types from Question 1.8, which use the Pin Electronics?

NOTES:

Analog Signals

Analog devices are also easy to classify (sort of). A device can be considered purely analog if it has no digital signals. Analog signals can have, at any instant in time, any one of an infinite number of levels. This is not a theoretical infinite—a real infinite amount of signal levels exists within any defined minimum and maximum signal points. To illustrate this, consider two whole integer values, e.g. 1 and 2. Between 1 and 2 are fractional values; how many fractional values exist between 1 and 2? Since there is an infinite amount of whole numbers and a fraction can be formed by dividing any whole number into 1, there is an infinite amount of fractions between 1 and 2. The same is true between 3 and 4, 1 and 4 or any two values. (For the curious, this leads to the interesting conclusion that the infinity of fractions is infinitely bigger than the infinity of whole numbers.)

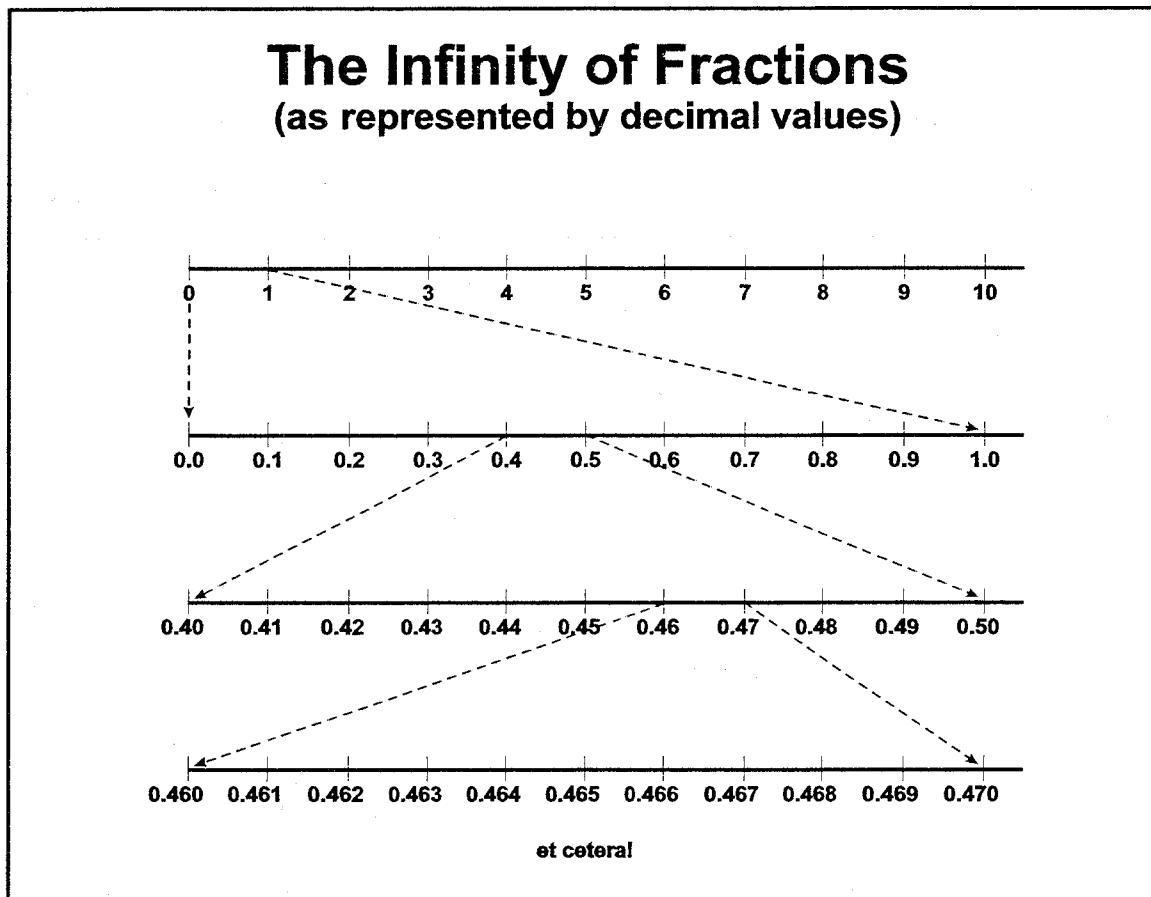


Figure 1.11—The infinite possibilities of dividing by 10

NOTES:



The infinite levels in an analog signal are a lot more than the 2 in a digital signal and one of the reasons an analog signal can contain a lot of information in a little bit of signal. Unfortunately, it also makes creating, analyzing and using analog signals much more difficult than digital signals. Without getting too far into semantics, a digital signal is really a special, limited case of analog signal.

Analog signals are generally considered to be *continuous* in that there are no abrupt changes in signal level, although this is not always true. The primary use of an analog signal is similar to that of a digital signal—to represent information. The information can be encoded in many ways: e.g. as a level, as a variation in frequency (frequency modulation), amplitude (amplitude modulation or AM) or as digital data (Manchester coding).

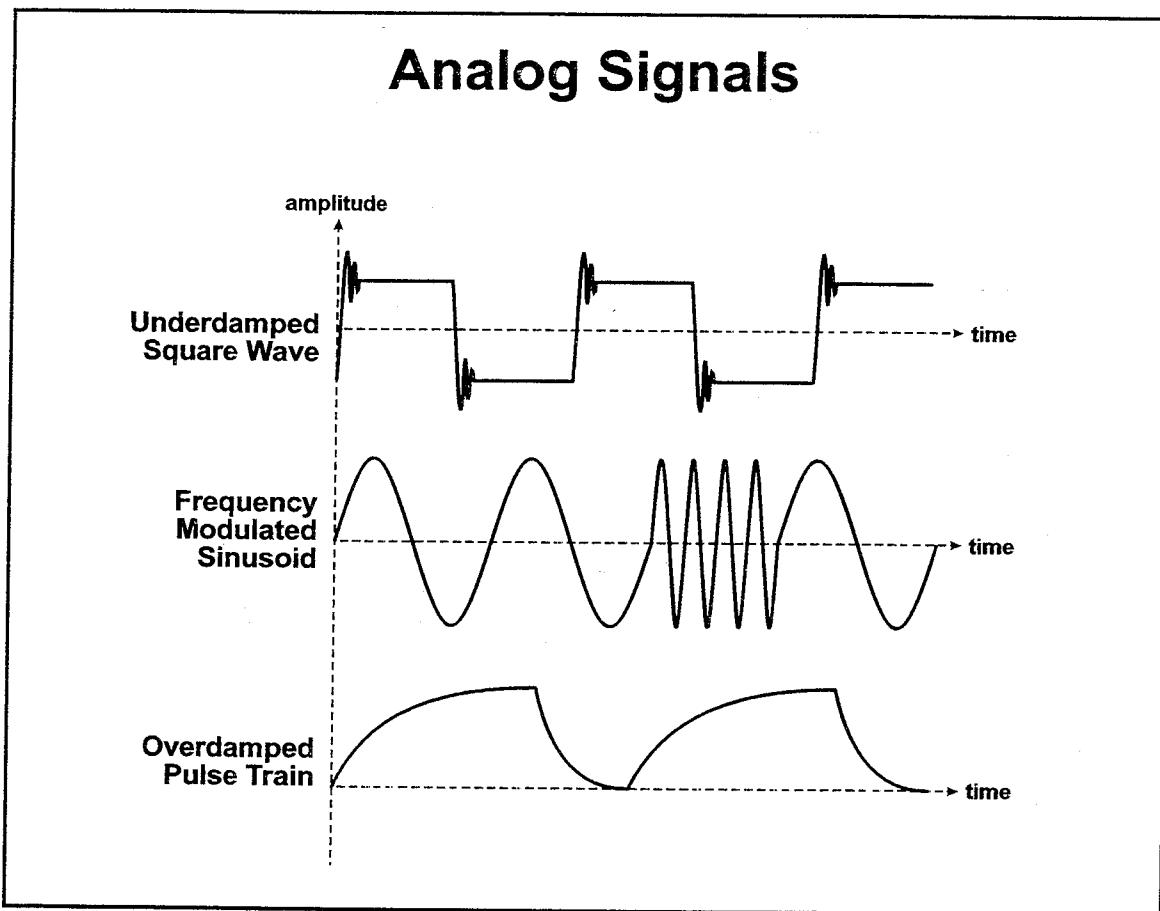


Figure 1.12

NOTES:

The Time aspect of analog signals

Unlike digital signals, analog signals contain no information without time. A digital signal is off or on, even if it never changes. An analog signal which does not change with time imparts essentially no information. (Consider a thermometer which never changes its temperature reading.) Even the most basic of analog signals, sound waves, have at least one frequency. The primary use for a DC analog signal is as a power supply, which provides no information, only power.

It is the changing of analog signals that allows us to encode information in them. A simple amplitude modulated sound wave tells us that an ambulance approaches. More complex modulation and encoding schemes allows huge amounts of information to be encoded into signals which extend into the tens of GHz in frequency.

NOTES:

Analog Circuits

The category of analog circuits includes virtually any component type or combination which is not a digital circuit. Familiar circuits for processing analog signals are amplifiers, filters, mixers (modulators), multipliers, sum or difference circuits, integrators, oscillators and comparators. (A comparator is typically considered an analog circuit, although its output has only 2 levels and is closer to a mixed signal circuit.) Analog circuits do not always require active circuits—many useful circuits (e.g. filters, attenuators) are possible with the use of only resistors, capacitors and/or inductors. And don't forget the ubiquitous on/off switch.

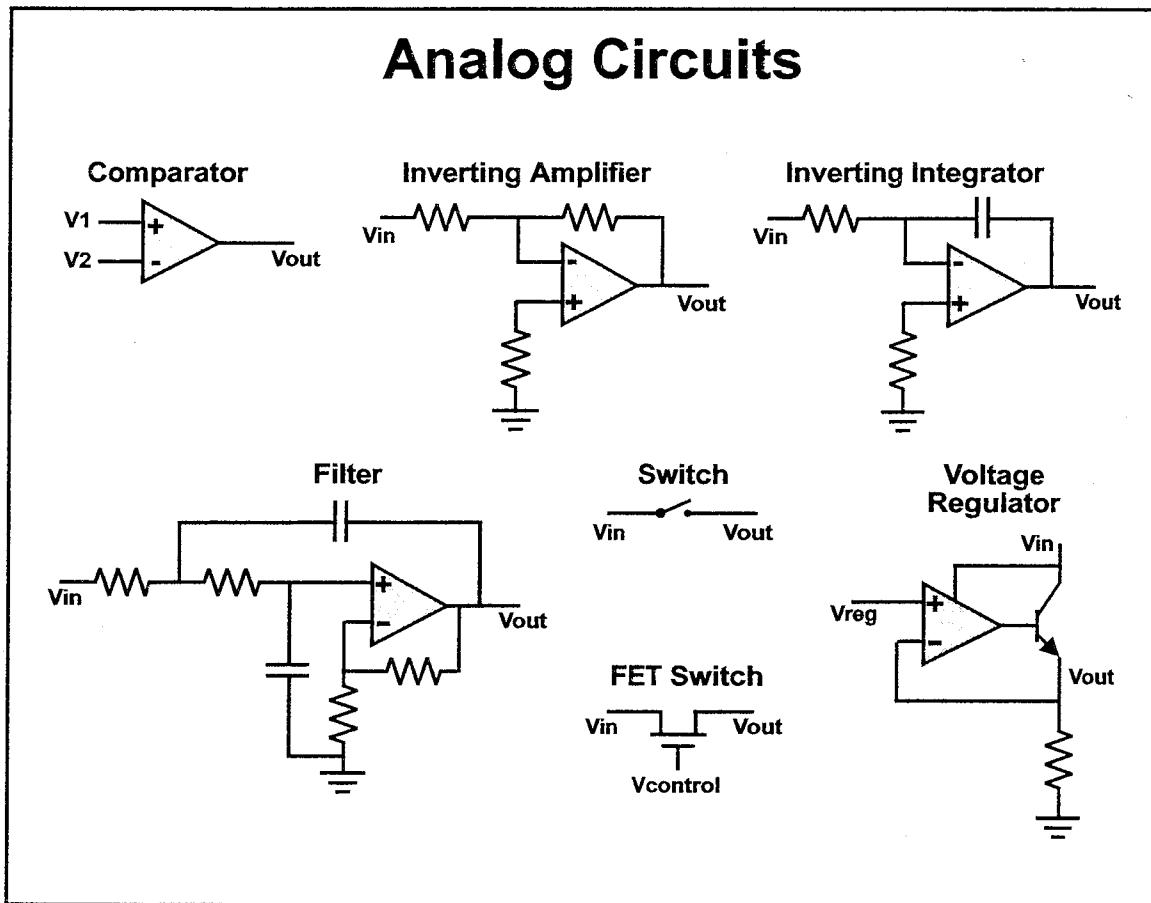


Figure 1.13

NOTES:

Overview of Mixed Signal Testing

The basic analog building blocks do mathematical operations on signals, e.g. addition, subtraction, multiplication, division, integration, differentiation. This is why they were named *analog* signals, because they were used to mimic the behavior of physical systems, thus the signals are *analogous* to the real world. Analog computers were used to model such systems as automobile shock absorbers or spring tensioners in which the solution of differential equations is required.

Analog circuits are often called "linear" circuits. There are different semantics in the use of the word "linear." One is for a signal that meets the strict mathematical definition of a linear function—its dependent variable (i.e. the circuit output signal) has no terms which are a power higher than 1. Another is the mathematical definition for a device output/input transfer function in which superposition holds, i.e. the output is a summation of terms of all input signal components times the transfer function. A third is that of a circuit element that has a straight line as its transfer element, e.g. a resistor has a "linear" voltage-current relationship and a diode has "non-linear" relationship. For this reason, we prefer the phrase *analog* rather than *linear* to describe all components and subcircuits that are not digital. Stated another way, we consider a circuit that has more than 2 signal levels an analog circuit.

Question 1.11: a) Can there be an analog circuit which does not require a power supply? b) A digital one?

Question 1.12: How can a switch be considered an analog circuit?

NOTES:



Analog Test Systems

Unlike relatively new digital test systems, analog test systems have a long history. They go back to the early days of General Radio Corp.'s manufacture of LRC (inductance, resistance, capacitance) bridges for measurement of those basic component properties and d'Arsenal meters for voltage and current measurement.

"Rack and Stack"

Traditional automation of analog testing was done with groups of individual instruments such as voltmeters, power supplies and signal generators placed in a rack and stacked on top of each other, thus their moniker *rack and stack*. Rack and stack systems use a computer to control and coordinate the various instruments and may be connected through various software protocols such as IEEE-488 (also called GPIB or HPIB), VXI (an extension of VME interface), or RS-232 serial interface. Rack and stack systems are still used fairly extensively for testing devices whose test requirements exceed the abilities of available commercial ATE (automatic test equipment) systems.

Traditional Analog ATE

Minicomputer based analog ATE became available as integrated systems whose test instruments were combined into a standard enclosure and memory mapped into the computer's I/O space. In other words, the minicomputer addresses the instruments directly instead of having to go through a special protocol like RS-232. This allowed creation of specialized software to make the instruments more uniformly programmable. Of course, different manufacturers did not have compatible software but the software within a given system was more consistent.

NOTES:

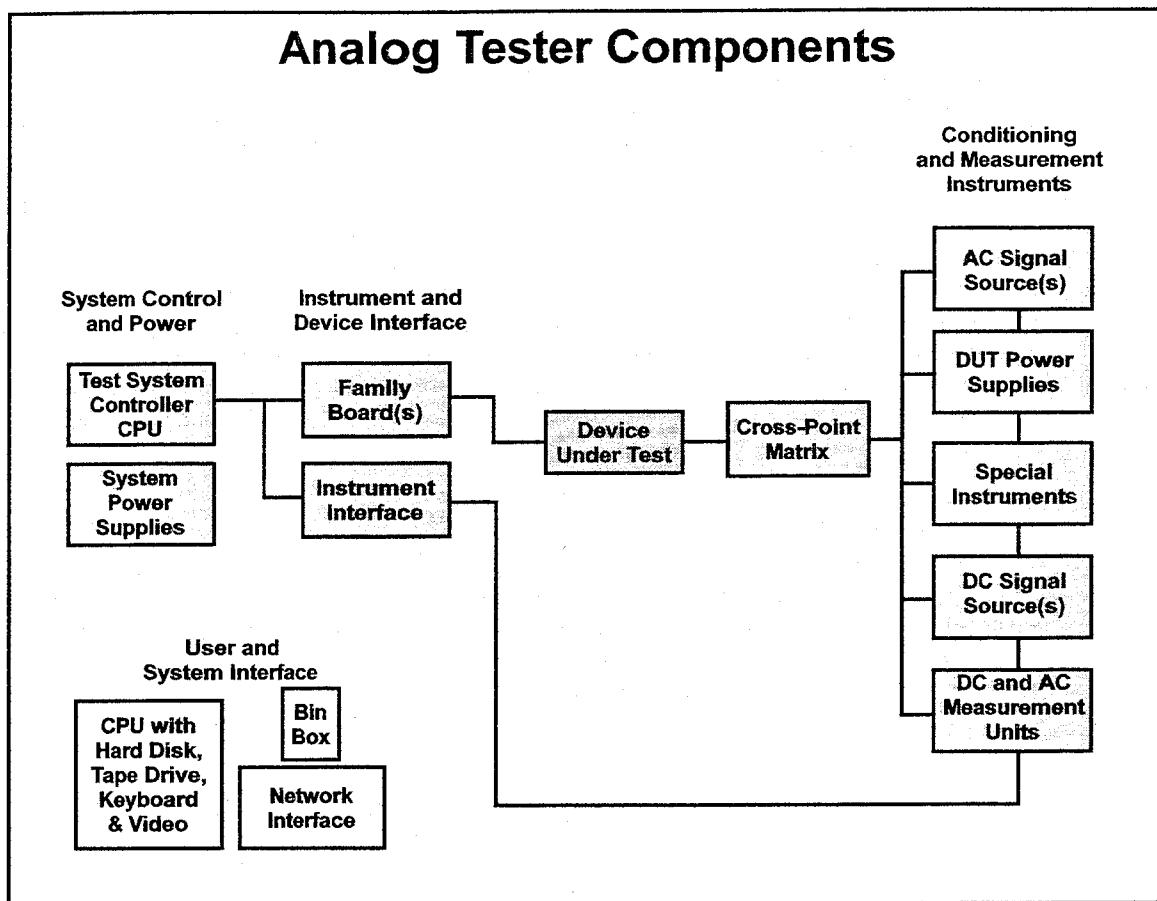


Figure 1.14

These ATE systems contain automated versions of traditional analog test instrumentation such as integrating voltmeters, function generators, filters, AC meters and often a chunk of custom circuitry at the DUT (device under test) site. Because of the limited quantity of instruments, a relay matrix is used to connect various DUT pins to different instruments at different times. The relay matrix, sometimes called a *cross-point matrix* because of its ability to cross connect any X point to any Y point, (see Figure 1.15) is set in software to connect device pins to the force and measure instruments to measure a specific parameter. Generally speaking, use of these instruments results in a long test time, one of the primary drawbacks of traditional analog ATE.

NOTES:

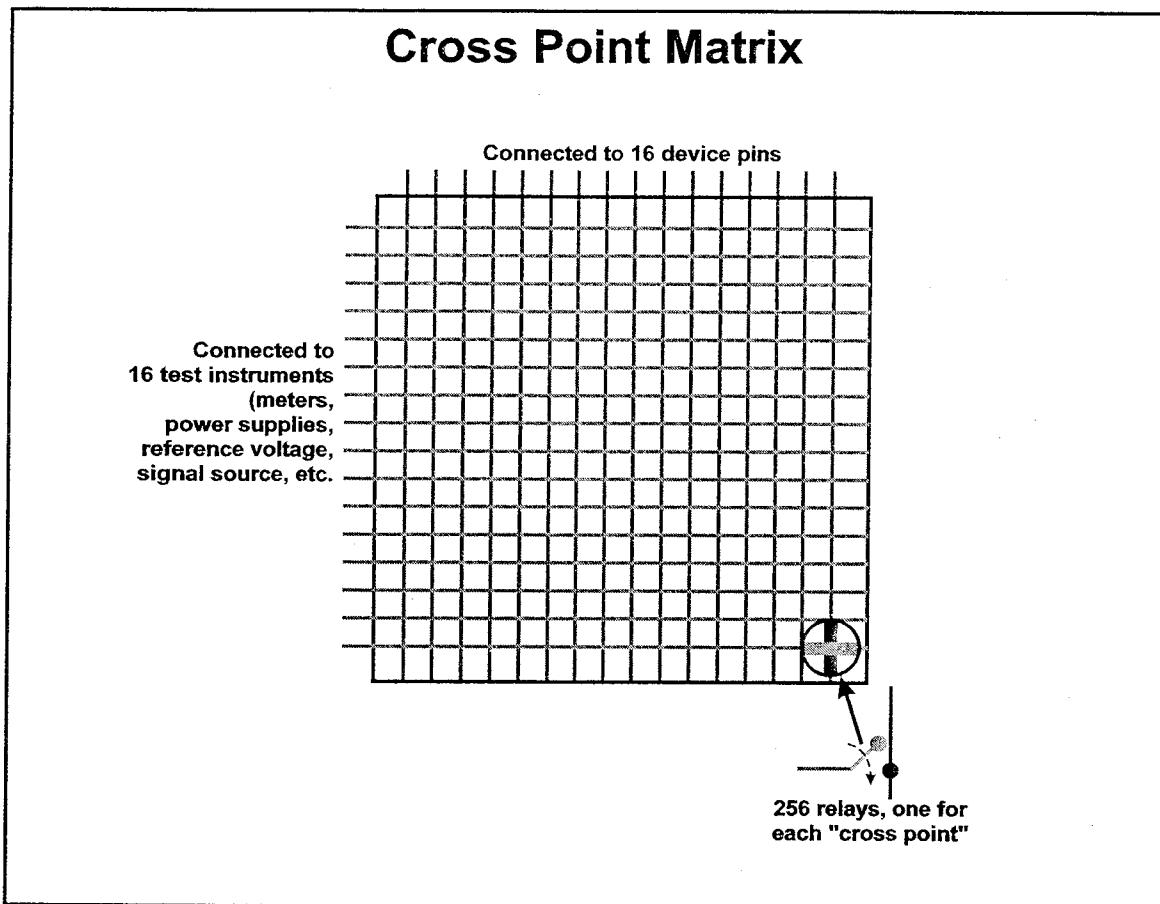


Figure 1.15

Just as analog circuits are widely varied in their functionality, analog test systems have various specialties. There are test systems tailored toward power devices such as IGBT and FET switches, high frequency devices such as RF receivers and *jelly bean* devices such as op amps and amplifiers. They have test sites and instrumentation especially configured for certain device categories. Naturally a system designed with 100mA relays in its cross point matrix would not last long testing 10 amp FET switches. Some companies still build their own rack and stack systems and write their own software, some out of necessity due to test requirements and some because they justify it on a cost basis (although the cost of software is often underestimated).

NOTES:

Device specific test system hardware and software are often called *family boards* or *device resources*. For example, op amps require a special feedback loop test configuration to test many of their parameters. An op amp family board consists of a set of hardware for 4 loops so 4 op amps can be tested at once. This allows a quad op amp to be tested with a single insertion. This family board would contain the circuitry to perform various parametric tests, including various feedback elements, compensation, etc., as well as relays to switch among the circuitry.

Similar family boards can exist for other device types such as switched mode power supply controllers, voltage regulators, multipliers, demodulators, filters, etc. Often, test equipment manufacturers have a number of family boards for their ATE equipment for sale to test common device types. If a device is unique, a new family board must be created.

Question 1.13: What is the purpose of a "family board"? Give an example.

Question 1.14: The ability to connect a PMU to any pin on a digital tester is similar to what function on an analog tester?

Question 1.15: What is the primary drawback of traditional analog testers?

NOTES:

Mixed Analog and Digital Signals

Mixing analog and digital signals has huge benefits; for example digital processing of analog signals is a good way to store analog information. Digital signals are easy to store, whereas analog is not quite so easy and generally requires sophisticated encoding and decoding techniques. As is often the case, however, the mixed signal whole is greater than the sum of its analog and digital parts.

Both analog and digital circuits perform mathematical operations on real world signals. Analog circuits perform the math with dedicated hardware and process continuous signals. Digital circuits perform mathematical operations with what can be considered software, even though the software is sometimes converted into dedicated (digital) hardware to make it operate faster.

With fast, powerful digital circuits available, math operations can be performed without the physical constraints associated with analog hardware. If a filter can be modeled in software, a signal can be processed without the constraints imposed by the energy storage of analog capacitors and inductors.

If many signals must be processed at once, a bank of digital signal processors (DSP) can be used on a bank of signals and the separate DSP "devices" all match perfectly, unlike their analog hardware counterparts.

Instantly changing the center or attenuation frequency of an analog filter is virtually impossible, yet it's a simple matter of inserting a new algorithm into a DSP based digital filter. Creation of a digital version of a multiple sine wave tone requires only specifying the amplitude and frequency of each individual wave then performing an inverse Fourier transform. Making a digital calculation on a high frequency signal uses the same circuitry as that for a low frequency signal whereas the equivalent math for high versus low frequency signals in analog circuitry may mean changing from a 100 gram inductor to a 2 kilogram inductor.

Question 1.16: In what form does an inductor store energy?

- a. electric field
- b. marshall field
- c. magnetic field
- d. Gaussian field

Question 1.17: When is a DSP filter better than an analog filter? When is an analog filter better?

NOTES:

Mixed Signal Circuits

Mixed signal devices are those which mix analog and digital signals. The first and most obvious of the mixed signal devices (excepting the comparator mentioned earlier) were analog to digital converters (ADCs) and digital to analog converters (DACs). These broad categories encompass the devices which bridge the gap between the analog and digital worlds. Within these 2 categories are a lot of subcategories related to the type of analog signal to be converted, e.g.:

- ◊ the dynamic range of the analog signal (e.g. linear, logarithmic and auto-ranging converters)
- ◊ required digital resolution (e.g. 8 bits, 12 bits, etc.)
- ◊ required conversion accuracy (e.g. 0.1% of full scale range)
- ◊ required conversion speed
- ◊ analog signal boundaries (e.g. $\pm 5V$)
- ◊ required linearity

Some of the above subcategories could be considered differences in the specification, but in converter design there are many compromises between such things as speed and linearity, cost and resolution, etc. Those using data converters are not generally able to simply "put in a device with a better specification" as could be the case with an op amp.

Interestingly, some mixed signal devices have virtually no analog circuitry—they digitize an analog signal at the outset and do all their processing in the digital domain. If they mathematically process a numeric representation of the analog signal, they are still considered a mixed signal device. Similarly, a device which makes minimal use of digital data and primarily has analog functions is a mixed signal device (e.g. an analog multiplexer).

A combination of analog and digital circuits allows digital control of analog signals for routing (an analog switch, e.g. DG211, is a classic example), application of power, digital control of functional analog modules and lots more. Because human senses work almost exclusively with analog signals (the ternary traffic light being one exception), even in the most digital of systems there must be some sort of translation to analog if a human is involved.

NOTES:

In general, mixed signal circuits have a complex diagram so to show examples with an illustration is somewhat pointless. Instead a list of mixed signal devices is presented:

- ◊ ADCs
- ◊ DACs
- ◊ Analog switches and analog multiplexers
- ◊ Sample-holds and track-holds
- ◊ Switched mode power supply controllers
- ◊ Modems and codecs
- ◊ Disk drive controller circuits
- ◊ Switched capacitor filters
- ◊ Microprocessors with embedded ADC
- ◊ Ethernet circuits

Question 1.18: What is the definition of a mixed signal circuit?

Question 1.19: Does a mixed signal circuit require an ADC or DAC?

NOTES:

Mixed Signal Test Systems

In the mid-1980s, with the cost to fabricate a silicon device decreasing rapidly, device testing became a significant portion of manufactured cost. Also, with customers demanding higher quality devices, more thorough testing was becoming necessary. Because test time is the biggest contributor to test cost and more thorough testing means longer test times, companies began to investigate ways to reduce test time. The most important technique to emerge from this research was the use of digital signal processing (DSP) to analyze and generate analog signals via digital data. This requires 3 special instruments: a signal processor, a waveform generator and a waveform digitizer.

A signal processor is a digital calculating machine which is specialized for processing arrays of numbers. (DSPs are also called *array processors*.) These arrays hold digital representations of numbers, often a *complex* number containing a *real* and *imaginary* part (more about this later). At this point in our discussion, we will only note that an array processor does the mathematics required to analyze the time and frequency characteristics of analog signals. It does its job much faster than the analysis done by traditional instruments such as DVMs or analog filters.

The presence of an array processor indicates that we are using a digital representation of analog signals, which leads us to the need for a waveform digitizer (WD). This instrument takes *samples* of an analog signal and converts them into a digital value representative of the analog value at the instant the sample was taken.

To create any desired analog signal easily, a *waveform generator* (WG) is used. Rather than having a special signal generator for different requirements, a waveform generator takes a digital representation of a waveform and creates an equivalent analog signal. It is often called an *arbitrary* waveform generator (AWG) because of its ability to make virtually any arbitrary waveform you can dream up.

The description of a waveform digitizer sounds a lot like the description of an analog to digital converter and a waveform generator sounds a lot like a digital to analog converter. Because ADCs and DACs are the quintessential mixed signal devices, a mixed signal test system can be seen as adding a WD, a WG and a DSP to a digital test system. In fact, a good mixed signal test system is either designed from the start as such or has been redesigned in some aspect (e.g. different power supply routing, additional linear power supplies, better ground plane, high current wiring paths, etc.) to meet the requirements of analog signal measurements.

NOTES:

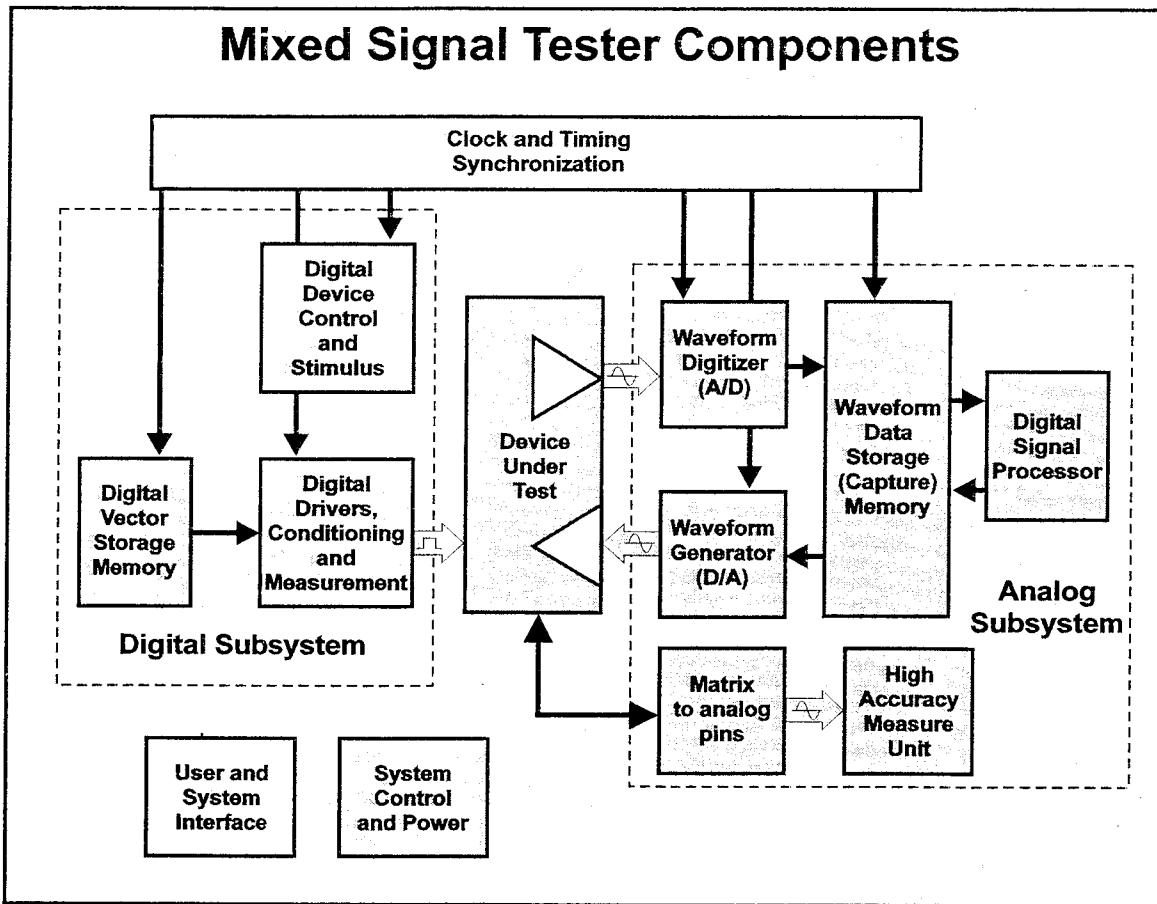


Figure 1.16

A mixed signal test system could be defined as one that tests mixed signal devices, one that contains digital and analog force and measure instruments, or some other way. ATE manufacturers all have their ideas of what makes a mixed signal test system. For our purposes, *a mixed signal test system is one that uses DSP techniques to process analog signals and uses parallel test vector techniques to process digital signals.*

The block diagram in Figure 1.16 shows the primary sections of a mixed signal tester defined this way. If necessary, review the prior discussion of digital and analog testers for information about a specific section. The various blocks in the digital subsystem come directly from Figure 1.3 and represent vector memory, pin electronics and other resources of a digital test system. The analog subsection contains a cross-point matrix and a high

NOTES:

Overview of Mixed Signal Testing

resolution measurement unit as discussed for a traditional analog test system and the 3 new resources just mentioned. These new resources replace many traditional analog tester instruments which each had a narrowly defined purpose with 3 general purpose resources and some memory storage.

A modern mixed signal tester as seen in Figure 1.15 should be able to test devices as purely digital as a microprocessor and as purely analog as an operation amplifier. They should also be able to test mixed signal devices such as those discussed in the section *Mixed Signal Circuits* on page 33.

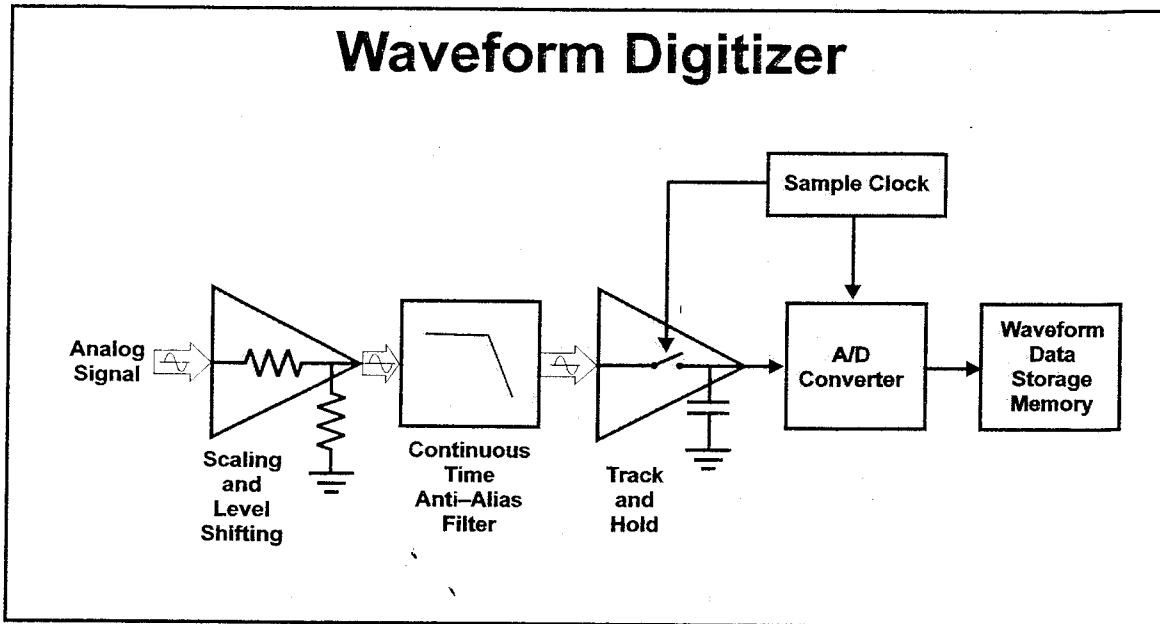
Waveform Digitizer

A waveform digitizer takes samples of analog signals and stores them in memory (see Figure 1.17). The major blocks of a waveform digitizer are signal conditioning circuitry, a track and hold, an ADC and data storage memory (often called *capture memory*). Of these, the signal conditioning is often the one you have the most control over and the one that can have the most direct impact on the accuracy of your test results.

The track and hold or the ADC input of a test system may have requirements which are not met by an incoming analog DUT signal, such as a limited input voltage range or a limited common mode signal range (in the case of a differential signal). The signal conditioning circuitry allows the incoming analog signal to be modified so that it fits these restrictions. For example, if the analog signal has a range centered around 5V, it can have 5V subtracted from it by a differential amplifier so that it is centered around 0V; if it has a 20V range but needs a 10V range, it can be scaled.

Conditioning also requires filtering to prevent the problem in digitizing known as *aliasing*. Any frequency components in the incoming signal which are above $\frac{1}{2}$ the sample clock frequency must be removed by the filter. Test systems often have banks of filters from which an appropriate one may be selected via software. In other situations you will need to place a filter on the load board. Note that this must be a *continuous time* (or "old fashioned") analog filter; a digital filter is not acceptable as it adds, as well as removes, higher frequencies.

NOTES:

**Figure 1.17**

The track and hold circuit does 2 things. First, it takes a high speed snapshot of the filtered analog signal at each sample point. ADCs take a long time to convert a signal by comparison to the track/hold so a track/hold allows much faster moving signals to be digitized.

Second, if the signal changes while the ADC is making its conversion, the conversion result is invalid. The track/hold holds the analog signal at a constant level while the ADC does its job. Finally, the ADC digital output word is stored in the capture memory so it can be analyzed by the DSP. Some test systems have multiple WDs that can synchronously sample more than one signal.

Waveform Generator

A waveform generator is a test system component with memory for storing data which represents the signal to be generated, a DAC to create an analog signal from the data and possibly some conditioning circuitry. The generator may have additional components which allow for modulation, output offset, differential signal generation, etc. Any wave shape can be generated, within the limits of the maximum conversion rate and voltage limits of the DAC and associated circuits.

NOTES:

A set of data points for a given wave shape is stored in the Waveform Data Storage Memory. Each time the Sample Clock occurs, a new data point is passed into the DAC, changing its output signal to the new value. This output signal may be conditioned by adding a DC offset, by modulating it with another analog waveform or by filtering it. Filtering is almost always required because of the "stepped" nature of the output signal as the DAC changes from one point to the next. The minimum step size of the generator is the same as the LSB size of a DAC. This LSB size may be adjustable by changing the Full Scale Reference signal into the DAC or by attenuating the DAC output signal (not shown in figure).

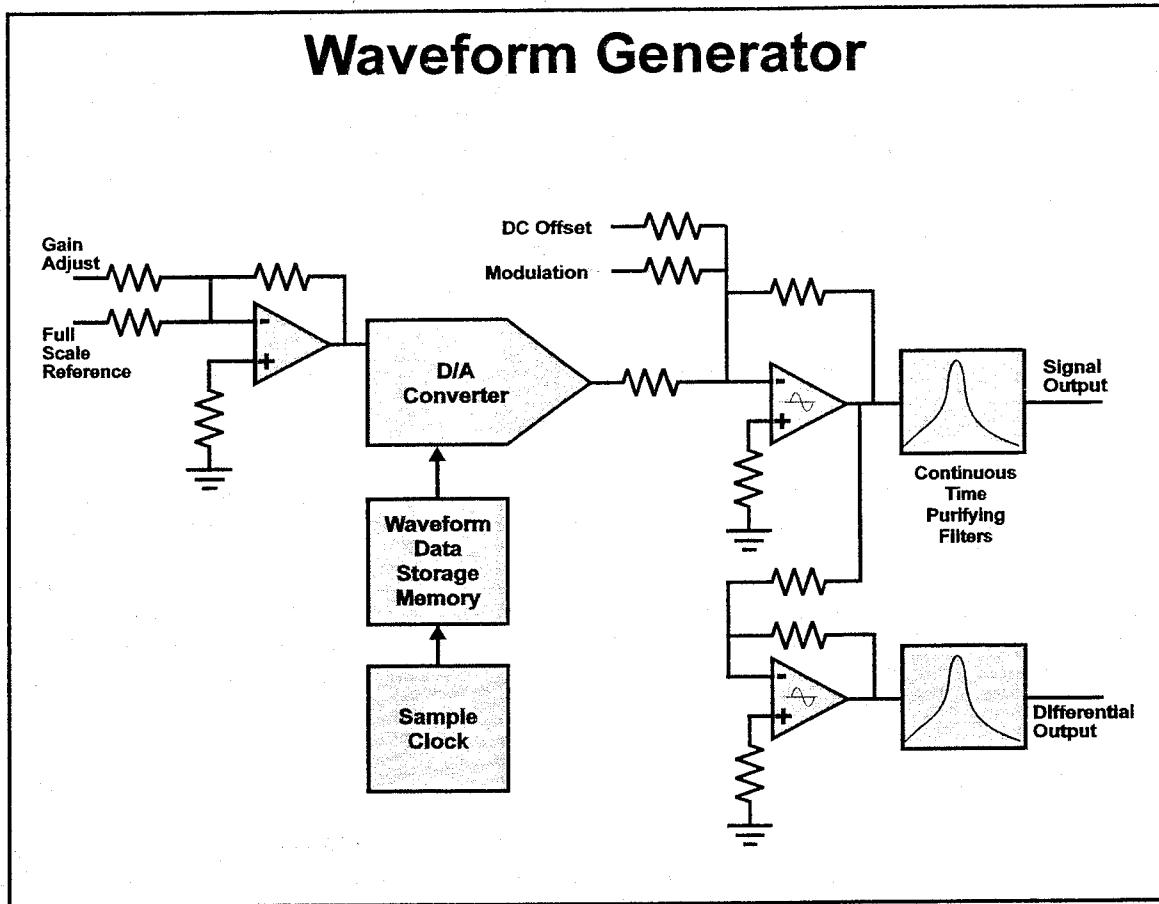


Figure 1.18

NOTES:



All test systems may not have all the components shown in this diagram or they may have more. For example, some testers have an additional DAC to correct the linearity of the primary DAC for generating high accuracy signals.

Some test systems also have an addition to the waveform storage memory which allows it to be sequenced. This allows sections of a stored waveform to be repeated or different waveform pieces to be concatenated into a more complex waveform.

Test systems will often have multiple waveform generators which are specialized for different purposes. For example, there may be a low speed high accuracy generator and a higher speed less accurate generator.

Waveform Data Storage (Capture) Memory

The digital representation of a waveform captured by a WD or about to be generated with a WG must exist somewhere; this is the place. This RAM is often called *Capture RAM* because it is used to capture waveform data as it comes from the WD, although it can also hold data for waveforms to be created with the WG.

Capture RAM has a high speed data bus connection to the DSP which allows the DSP to operate at its absolute maximum speed. As a "number cruncher," the DSP must be able to get numeric data into and out of its registers quickly; the Capture RAM makes certain that this is possible. Capture RAM can sometimes be used for other tasks, such as rearranging data words that were not digitized in the best sequence for DSP manipulation.

Digital Signal Processor

The DSP is, in essence, a microprocessor specialized for performing mathematical operations on arrays of digital numbers. There are many known algorithms for processing numeric time sampled data into frequency and/or phase data. Among these algorithms are the Discrete Fourier Transform (DFT) and various forms of the Fast Fourier Transform (FFT). The architecture of a DSP is optimized to allow fast multiplication and summing, logarithm calculations, squaring and square root calculations and more, in addition to the fast data I/O mentioned above. The DSP usually contains or accesses special high speed RAM to hold input data, intermediate calculation data and output data. Often a test system can do DSP calculations and capture DUT data simultaneously to decrease test time.

The DSP functions most often used in device testing are available as predefined functions on most mixed signal test systems. These generally include functions for calculating THD, SNR, SINAD, signal power, conversion to dB, time domain to frequency spectra conversion, frequency spectra to time domain conversion, *effective number of bits* calculation, *envelope detection*, summing, differencing, multiplying or dividing 2 data sets, and more. See your test system software manual for available "canned" algorithms. And remember that the result of a calculation is only as good as the data you collect!

Question 1.20: What is the definition of a mixed signal tester?

NOTES:

Software Overview of Mixed Signal Testing

Question 1.21: What are the 3 important new instruments used by a mixed signal test system to replace many traditional analog tester instruments?

Answer: Waveform digitizer, Logic Analyzer, Vector Network Analyzer

Question 1.22: Can a waveform digitizer and a waveform generator use the same capture memory?

✓

Question 1.23: What is the primary purpose of Capture RAM?

Answer: To store captured waveforms for later analysis

Question 1.24: Is Capture RAM the same as vector memory?

Answer: No, Capture RAM is used for waveform storage, while vector memory is used for signal processing.

Question 1.25: Why would you want to digitize two signals at the same time?

Answer: To analyze and correlate the two signals simultaneously.

NOTES:

Key Points of This Chapter

- ◊ Digital signals have 2 levels.
- ◊ Digital signals store information more easily than analog signals but store it less densely.
- ◊ Digital devices can be simple or complex, but their specifications are very similar (for a given process technology). It is primarily their vector patterns that are different.
- ◊ Digital test systems use a standard set of hardware and software resources that can apply to many different digital devices, as long as maximum frequency and timing conditions are met.
- ◊ Analog signals have infinite levels.
- ◊ Analog devices have many and varied purposes, and require many techniques for measurement of signals and calculation of limits.
- ◊ Analog test systems may have many specialized force and measurement instruments and may require different hardware for different device types (family boards).
- ◊ Mixed signal test systems are not a blend of traditional digital and analog test systems.
- ◊ Mixed signal test systems replace many analog test instruments with a waveform digitizer, a waveform synthesizer and a digital signal processor. This may also reduce the requirements for family board hardware.
- ◊ In general, mixed signal test systems perform analog measurements faster and with better repeatability than analog instrument based test systems.
- ◊ Mixed signal test systems are better than analog test systems for testing many purely analog devices.
- ◊ You need to understand the functions of a traditional digital test system to use a modern mixed signal test system.
- ◊ Mixed signal test systems require signal conditioning for analog DUT signals being digitized and analog tester signals being generated and sent to the DUT.

NOTES:

Answers to Chapter Questions

Answer 1.1: The contain only 2 signal levels and can be easily stored.

Answer 1.2: A bit.

Answer 1.3: Data storage, signal flow control, calculations.

Answer 1.4: Floating point and integer.

Answer 1.5: $2^{16} = 65536$

Answer 1.6 : b. Noise margin

Answer 1.7: The test must verify that the device works at the worst case requirements, not the best case or nominal requirements.

Answer 1.8: DC (or static) tests, AC (or dynamic) tests and functional tests

Answer 1.9: DC tests use the PMU

Answer 1.10: All use the pin electronics

Answer 1.11: a) Two examples are a volume control (attenuator) and a passive (LC, RC or LRC) filter. b) No.

Answer 1.12: By the definition given in this chapter, an analog circuit is one that can have more than 2 signal levels. An analog switch is a circuit which connects or disconnects an analog signal. Even though it has a binary "on/off" function, its primary purpose is to allow or restrict an analog signal's passage.

Answer 1.13: It has test circuitry that is common to similar device types, e.g. a family board could test single, dual or quad op amps.

Answer 1.14: A cross point matrix.

Answer 1.15: Using traditional sources and measurement techniques with a cross point matrix generally means a very slow test.

Answer 1.16: c. Magnetic field.

Answer 1.17: This is a question to provoke thought. If no digital subsystem or DSP is available, clearly an analog filter is the only choice. If the speed required for calculation exceeds the capabilities of the DSP in use (e.g. processing GHz signals), an analog filter is required.

When a DSP subsystem is available or can be designed in and the calculation speed is not too great, digital filtering is the best choice.

NOTES:



A good example is cellular telephones—their circuitry for linking to a repeater or a satellite runs at MHz to GHz frequencies and is analog. The audio voice encoding, decoding and filtering requires only a couple of thousand Hertz of analog bandwidth and is done with DSP.

Answer 1.18: A circuit that operates on both analog and digital signals.

Answer 1.19: No; consider a switched mode power supply controller or an analog mux.

Answer 1.20: One that uses DSP techniques to process analog signals and uses parallel test vector techniques to process digital signals.

Answer 1.21: Waveform digitizer, waveform generator and digital signal processor.

Answer 1.22: Yes, although it will depend on the specific test system whether they do or not.

Answer 1.23: To hold WD and/or WG data for the DSP and pass it to and from the DSP so it can operate at maximum speed.

Answer 1.24: No; the two types of memory are very different.

Answer 1.25: A couple of possible reasons: 1. to save test time 2. to see how signals are related in time

NOTES:

References

1. *The Fundamentals of Digital Semiconductor Testing*, Soft Test Inc., 246 Union Ave., Los Gatos, CA 95132, 1998.

NOTES:

Basic Analog Specifications

Goals

- ◊ Review basic analog device specifications for op amps, comparators and filters
- ◊ Understand how these parameters can affect test results, due to DUT and external conditioning circuitry

Objectives

- ◊ To provide a baseline knowledge of the types of specifications required to do DSP based testing
- ◊ Gain familiarity with fundamental specifications of analog circuits
- ◊ Understand how these specifications relate to mixed signal testing and to mixed signal devices

What you will learn

- ❖ Op amp, comparator and filter specifications

NOTES:



Digital Device Specifications

Digital devices have a common set of performance specifications that make a large portion of their test specifications generically similar. These specs are the worst case input and output voltage and current levels, power supply current and the time specifications of propagation delay, setup time and hold time. Different devices will have different values for these specifications (especially for different manufacturing technologies e.g. CMOS vs. ECL) but the concepts behind the specifications and methods for testing them are the same. The other primary difference between different digital devices is the truth table or, in ATE parlance, the test vectors. Following is a list of digital test parameters:

- ◊ VOL and VOH—Logic low and high output voltage levels
- ◊ VIL and VIH—Logic low and high input threshold voltages
- ◊ IIL and IIH—Logic low and high input currents
- ◊ IOL and IOH—Logic low and high output currents
- ◊ IDD—Power supply current
- ◊ IOS—Maximum output current
- ◊ IOZ—Output high impedance current
- ◊ f_{max} —Maximum operating frequency
- ◊ t_{su} —Setup time
- ◊ t_h —Hold time
- ◊ t_{pd} —Propagation delay
- ◊ t_{en}, t_{dis} —Three state enable/disable time

Some of these parameters are tested statically by making a DC measurement (e.g. VOH) and some are tested dynamically by making high speed voltage comparisons while running a vector pattern (e.g. VIL or f_{max}) or by making an average DC measurement while running a vector pattern (e.g. Dynamic IDD).

These specifications are not discussed in detail in this course. They are discussed in *The Fundamentals of Digital Semiconductor Testing*¹; a class is available in this subject from Soft Test Inc. See the information at the front of this book.

NOTES:

Analog Circuit Specifications

It is evident from Figure 1.13 on page 23 that the basis for many analog circuits is the operational amplifier, or *op amp*. Thus our discussion of analog device specifications begins with that device. It is impossible for this course to cover all specifications of all analog device types, so those discussed here will be the ones which apply most generally to many device types.

Op Amp specifications

An op amp is an amplifier whose output is based on the difference between its input signals. It is used almost universally in a negative feedback configuration. The effect of negative feedback is to cause the output to drive the signal at the negative input so that it is equal to the voltage at the positive input. (We are discussing the classic *voltage feedback* or *transconductance* configuration here, not current feedback. This is an op amp with high input impedance and low output impedance.) The following parameters are discussed:

- ❖ Input offset voltage (V_{IO})
- ❖ Input bias current (I_{IB})
- ❖ Input offset current (I_{IO})
- ❖ Common mode rejection ratio (CMRR)
- ❖ Power supply rejection ratio (PSRR)
- ❖ Gain bandwidth (GBW)
- ❖ Total harmonic distortion (THD)
- ❖ Noise
- ❖ Signal to noise ratio (SNR)
- ❖ Signal to Noise and Distortion (SINAD)
- ❖ Slew rate (SR)
- ❖ Settling Time (t_s)

V_{IO} —Input offset voltage

The voltage required between the plus and minus inputs to force the op amp output to its zero (or null) voltage. Normally ranges from $\pm 10\mu V$ to $\pm 10mV$.

I_{IB} —Input bias current

The current required individually by each input to force the op amp output to its zero (or null) voltage. Can range from tens of pA to tens of μA depending on the design and fabrication technology of the specific op amp. Can often be compensated by having equal resistance to both op amp inputs.

NOTES:



I_{IO}—Input offset current

The difference between the 2 input bias currents. Cannot be compensated but it is generally an order of magnitude less than I_{IB} .

CMRR—Common mode rejection ratio

A measurement, usually in dB, of the ratio of signal input to output with the same signal applied to both positive and negative inputs. Ideally, this parameter is infinite, so if you connect an op amp with both inputs tied together the output remains at zero (or null voltage). Real devices have CMRR on the order of -100dB at DC. But beware! CMRR often decreases rapidly with increasing input frequency. Given an output signal V_{out} and input signal V_{in} , CMRR is calculated as:

$$CMRR = 20\log\left(\frac{\Delta V_{out}}{\Delta V_{in}}\right) \quad (2.1)$$

where Δ represents a change in voltage.

PSRR—Power supply rejection ratio

A measurement, usually in dB, of the ratio of output signal change due to a change in power supply voltage(s). Ideally, this parameter is infinite, so power supply fluctuations have no effect on op amp output. Real devices have PSRR on the order of -100dB at DC. But beware! Like CMRR, PSRR often decreases rapidly with increasing input frequency. Given an output signal V_{out} and power supply signal V_{ps} , PSRR is calculated as:

$$PSRR = 20\log\left(\frac{\Delta V_{out}}{\Delta V_{ps}}\right) \quad (2.2)$$

where Δ represents a change in voltage.

GBW—Gain bandwidth

The frequency at which the gain of an op amp drops by 3dB below its value as measured at DC. In general, the gain bandwidth product is a constant, so bandwidth is a maximum at unity gain (excepting gain < 1, in which case an op amp is probably not necessary).

Noise

Random signals created within the op amp caused by any of several physical phenomena such as electrons rearranging themselves (*thermal* or *Johnson* noise), stored charge moving about within internal device points

NOTES:



(low frequency, $1/f$ or *flicker noise*) or internal particle movement due to current flow (*shot noise*). Noise is specified in $V/\sqrt{\text{Hz}}$ (volts per root Hertz) or $I/\sqrt{\text{Hz}}$ (current per root Hertz).

As noise is a random effect, its amplitude is distributed evenly about 0 and its average value is zero. Also, noise is an AC parameter and must be measured with respect to frequency. Thus to measure the effect of noise, its power is measured, meaning the instantaneous voltage or current noise is squared. This power is measured over a frequency bandwidth, giving V^2/Hz or I^2/Hz , which is the *noise power spectral density*. To relate this back to voltage and current, take the square root to get $V/\sqrt{\text{Hz}}$ or $I/\sqrt{\text{Hz}}$. For an FFT spectrum, the total RMS noise over bandwidth for N samples and $N/2$ bins can be calculated with

$$V_{noise\,RMS} = 20 \log \left(\frac{\sqrt{\sum_{bin=1}^{N/2} V_{noise}^2}}{\sqrt{\frac{F_s}{2}}} \right) \text{db} \quad (2.3)$$

THD—Total harmonic distortion

The sum of all signals created within the op amp by nonlinear response of its internal forward transfer function. The frequency of harmonic distortion signals is always an integer multiple of the input test frequency. For example, if a 1KHz sine wave is the input signal, *even* harmonic distortion will occur at 2KHz, 4KHz, 6KHz... and *odd* harmonic distortion will occur at 3KHz, 5KHz, 7KHz...

Harmonic distortion is measured in dB as a ratio of the amplitude of the sum of harmonic signals divided by the input signal. In mathematical terms, given sinusoidal signal V_{sig} ,

$$THD = 20 \log \left(\frac{\sqrt{\sum_{k=2}^{\infty} V_k^2}}{V_{sig[rms]}} \right) \quad (2.4)$$

where V_{sig} is the RMS value of the signal of interest, and V_k is the RMS value of each harmonic frequency component from 2 times the frequency of V_{sig} to infinity. Consider each V_k term as the RMS voltage measured after filtering the signal to remove everything except that at k times the V_{sig} frequency, with $k = 2, 3, 4\dots$ Normally in

NOTES:



test situations, distortion past $n = 10$ is below the device noise level, so summing to infinity is not really necessary.

SNR—Signal to noise ratio

A ratio of the signal of interest to the total noise in a given bandwidth. This specification requires frequency spectrum information for both the signal and the noise. SNR is measured in dB as a ratio of the amplitude of the signal to the sum of noise. In mathematical terms, given a sinusoidal signal V_{sig} and a noise voltage V_n ,

$$SNR = 20\log\left(\frac{V_{sig[rms]}}{V_n[rms]}\right) \quad (2.5)$$

where V_{sig} is the RMS value of the signal of interest and V_n is the RMS value of the noise over the frequency bandwidth of interest. This parameter does not include harmonic distortion components.

SINAD—Signal to Noise and Distortion

A ratio which is a combination of THD and SNR, where the signal of interest is ratioed with the sum of all noise plus harmonic distortion.

SR—Slew rate

The maximum rate of change of the output signal given a stepped input signal. Specified in Volts/ μ sec and typically ranges from 0.1V/ μ sec to 100V/ μ sec depending on amplifier design and fabrication technology. This parameter normally requires the output to swing over most of its output range (e.g. -5V to +5V) to induce the output into its current limited mode.

t_s—Settling Time

Given a stepped input signal, the amount of time required for the output to reach and remain within a required accuracy band near the final output value. If you are using an op amp in a DUT test circuit, settling time can be especially important. Notice in Figure 2.1 that a signal can enter and leave the band between upper and lower settling limits, so measurement of settling time requires sampling the signal at a point where it is known to be settled, then “backing up” to the first point at which it leaves the settled band.

NOTES:

Settling Time

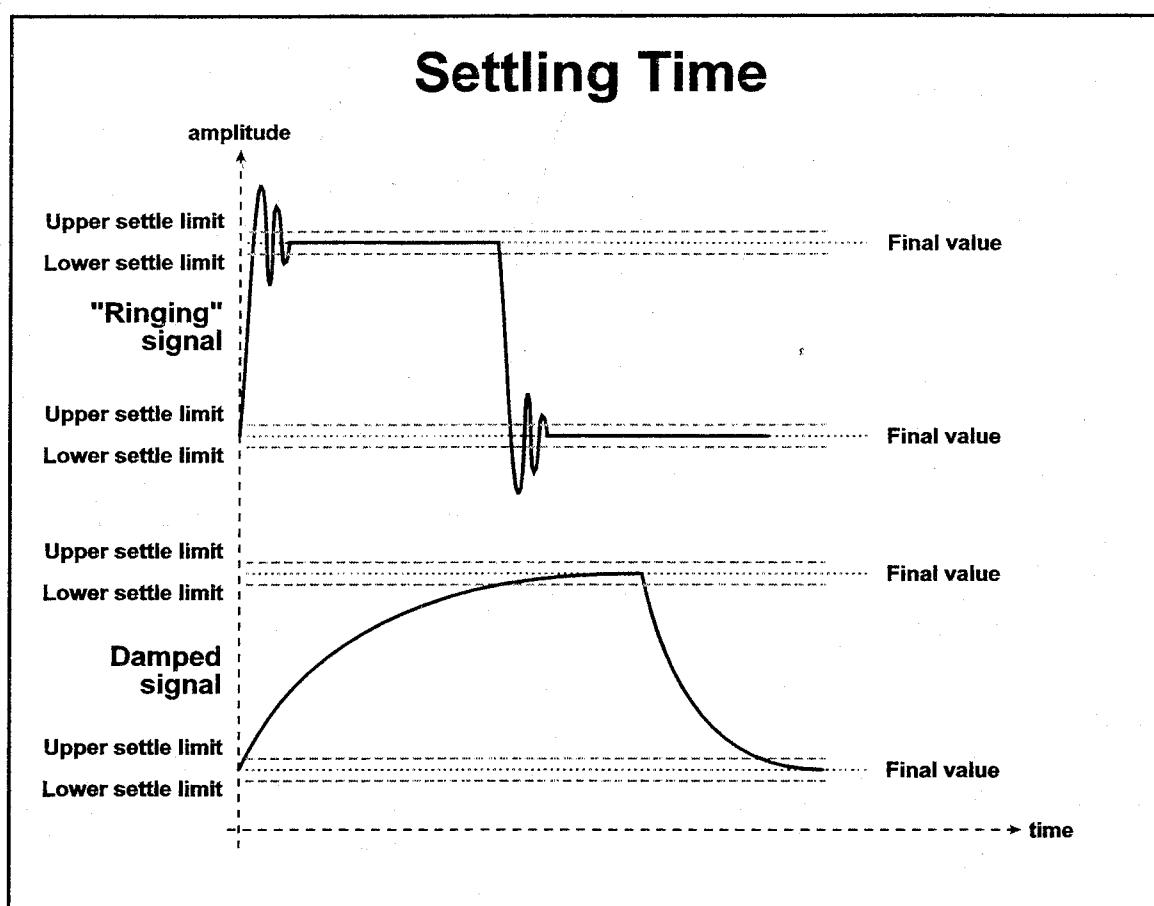


Figure 2.1

The initial point from which settling time is measured can be specified as when the input is changed, which means it includes input to output delay plus output slewing plus small signal settling. This parameter is normally measured with a small output swing (e.g. $\pm 100\text{mV}$) to prevent the output from going "non-linear", i.e. the output is not *slew rate limited*. It may also be measured from the point when it first enters the settled band until it remains stable within that band; this is primarily useful for measuring settling of a large scale signal swing which "rings."

NOTES:

Comparator Specifications

A comparator can be considered a mixed signal device, although it existed long before the phrase *mixed signal* was ever used. The output of a comparator is high when the voltage at its positive input is higher than the voltage at its negative input and low otherwise. An op amp with no feedback is a comparator, although not a very good one.

Comparators are optimized for fast operation and may have additional input signals to allow *gating* so its operation can be time synchronized with other circuitry. Gated comparators are used in digital functional testing to determine if a digital device's outputs are above or below VOL/VOH thresholds.

Comparators have many of the same input specifications as op amps, e.g. V_{IO} , I_{IO} and I_{IB} . The most important additional specification is:

Response Time

The time required for the comparator's output to switch from one state to another given a specified amount of *overdrive*. Overdrive is the difference in the two input signals, so a comparator will switch faster when one input is at 2V and the other is at 3V than it will if one input is at 2V and the other is at 2.2V. Response time on commercial comparators can range from tens of nsec on high speed devices to tens of μ sec on low power devices.

NOTES:



Filter Specifications

Filters are not discussed here from a perspective of how to test them, but because filters play a major role in successfully testing mixed signal devices. Examining a few of their specifications will help in future chapters.

Filters are strictly AC devices which allow certain frequencies to pass and reject other frequencies. Note that these specifications versus frequency refer to the frequency of a sine wave, as any practical signal can be represented by a summation of sine waves via a Fourier series. The following parameters are discussed:

- ❖ Bandwidth (BW)
- ❖ Quality Factor (Q)
- ❖ Center Frequency (f_0)
- ❖ Settling time

Bandwidth (BW)

The range of frequencies which are passed by a filter and attenuated by 3dB or less as compared to the *mid-band* or *center* frequency.

Quality Factor (Q)

A measure of the "sharpness" of the resonance (amplitude vs. frequency) curve. If a filter should only pass a single frequency, it requires a high Q. Q is defined as the center frequency divided by the bandwidth ($Q=f_0/BW$).

Center Frequency (f_0)

The frequency at which the filter's output signal is at its maximum amplitude. If the filter has a wide (or "flat") frequency response, f_0 is half way between the upper and lower -3dB points, i.e. at the center of the bandwidth.

Settling time

The time required for a filter output to stabilize after its input signal changes. Filters contain energy storage elements such as capacitors or inductors. When a filter's input signal is applied, the input signal has changed from a DC value of 0V to an AC value with an RMS component. This can be viewed as a form of input step change, which causes a transient at the filter output. The time required for this transient to settle is the filter's settling time.

This parameter depends on the center frequency and the *order* of the filter. (The *order* is an indication of the slope of the filter's amplitude vs. frequency curve.)

NOTES:

Filter Specifications

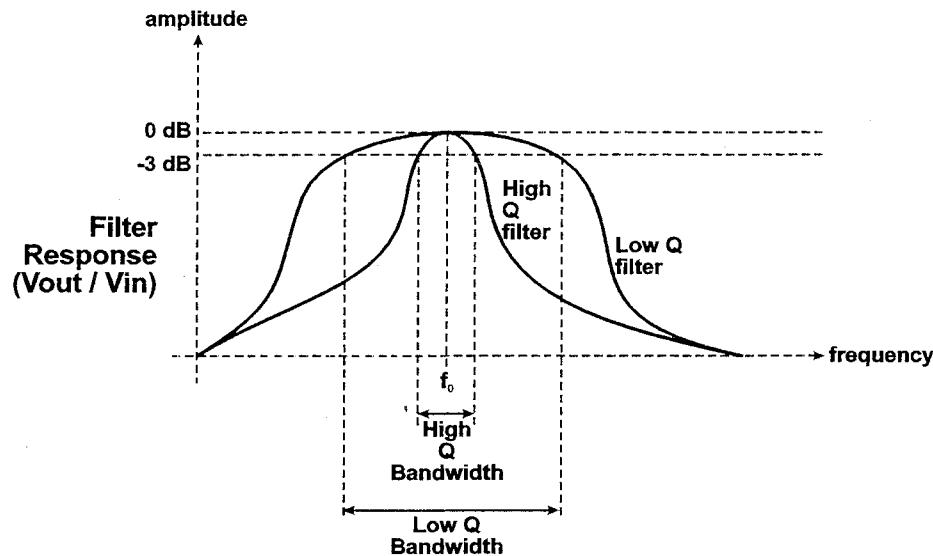


Figure 2.2

NOTES:

Key Points of This Chapter

- ❖ Digital device specifications apply to mixed signal devices.
- ❖ Analog device specifications apply to mixed signal devices.
- ❖ Mixed signal devices have their own special requirements and specifications above and beyond those of purely digital and purely analog devices. These requirements may mean that special measurement circuitry is necessary; proper configuration and use of the special measurement circuitry requires an understanding of basic analog specification parameters.

NOTES:



Answers to chapter questions

No questions in this chapter.

NOTES:

References

1. Guy Perry, *The Fundamentals of Digital Semiconductor Testing*, Soft Test Inc., 246 Union Ave., Los Gatos, CA 95132, 1998.

NOTES:

Digital to Analog Converter Static Measurements

Goals

- ◊ Examine DAC static specifications
- ◊ Explain measurement techniques for DAC offset and gain error
- ◊ Explain measurement techniques for DAC static parameters—DNL, INL
- ◊ Discuss various options for making low level measurements on an ATE system
- ◊ Straight line vs best curve fit INL

Objectives

- ◊ Have the student recognize the complex relationship between the ATE system and the DUT
- ◊ Enable the student to create the input stimulus necessary for DAC linearity testing
- ◊ Understand techniques of using a linearity reference signal
- ◊ Understand how to calculate DAC parameters such as LSB size, DNL, INL et al
- ◊ Understand how to make linearity measurements more accurate and repeatable
- ◊ Know where to look when test problems arise

What you will learn

- ❖ DAC static specifications—what they are and how they relate to DAC measurable quantities
- ❖ Test system configuration for DAC static parameter testing
- ❖ How to measure DAC zero and full scale
- ❖ How to calculate offset and gain errors
- ❖ How zero and full scale measurements are used to calculate linearity
- ❖ Techniques to make linearity testing faster and more accurate
- ❖ How to minimize the number of input codes necessary to test DNL/INL
- ❖ Possible error sources when measuring DNL/INL

NOTES:

Overview of Testing DAC Static Parameters

Before discussing how to test static DAC parameters, let's review these things:

- ❖ DAC static specification
- ❖ A general test system configuration for DAC static parameter testing
- ❖ An example data sheet
- ❖ A list of DAC static parameters and the measurements required to calculate their values
- ❖ A diagram of the errors

NOTES:

DAC Static Specifications

As the name implies, *static* DAC parameters are tested at a low test signal frequency, generally at DC. This implies setting a digital input code then measuring the resulting output signal value. In our examination, we discuss the static parameters for a DAC with a linear transfer characteristic. Important DAC static specifications are:

- ❖ Resolution
- ❖ Full scale range (FSR)
- ❖ LSB size
- ❖ Differential nonlinearity (DNL)
- ❖ Monotonic
- ❖ Integral nonlinearity (INL)
- ❖ Offset and gain errors
- ❖ Accuracy

Two other parameters of interest when testing DACs are

- ❖ Maximum conversion rate
- ❖ Settling time

Resolution

An indication of the smallest increment of its output range that the DAC can change. When given in bits, the resolution is the power of 2 which is the divisor for the full scale range to calculate the minimum DAC output change between successive input digital codes.

Full scale range (FSR)

The maximum extremes of output signal for a DAC. This parameter varies widely depending on the DAC and can be specified as current or voltage which is positive, negative or both. Common values are $\pm 2\text{mA}$, -2mA , $\pm 10\text{V}$, $\pm 5\text{V}$, $+1.024\text{V}$. Devices whose output does not cross through 0 are called *unipolar* while those with \pm output polarities are *bipolar*.

Offset error

The difference between the ideal and actual DAC output values when the zero or null level digital input code is presented to the device. This parameter can represent an end point code and, on bipolar DACs, can represent the mid-scale code, although this is rare.

NOTES:

Gain error

Gain error is the difference between the measured output when full scale input code is presented and the ideal full scale output. To make the gain error independent of offset error, the zero scale output of the device is considered as the zero or null point. The gain error is the full scale output measured value, minus the offset error, minus the ideal FSR.

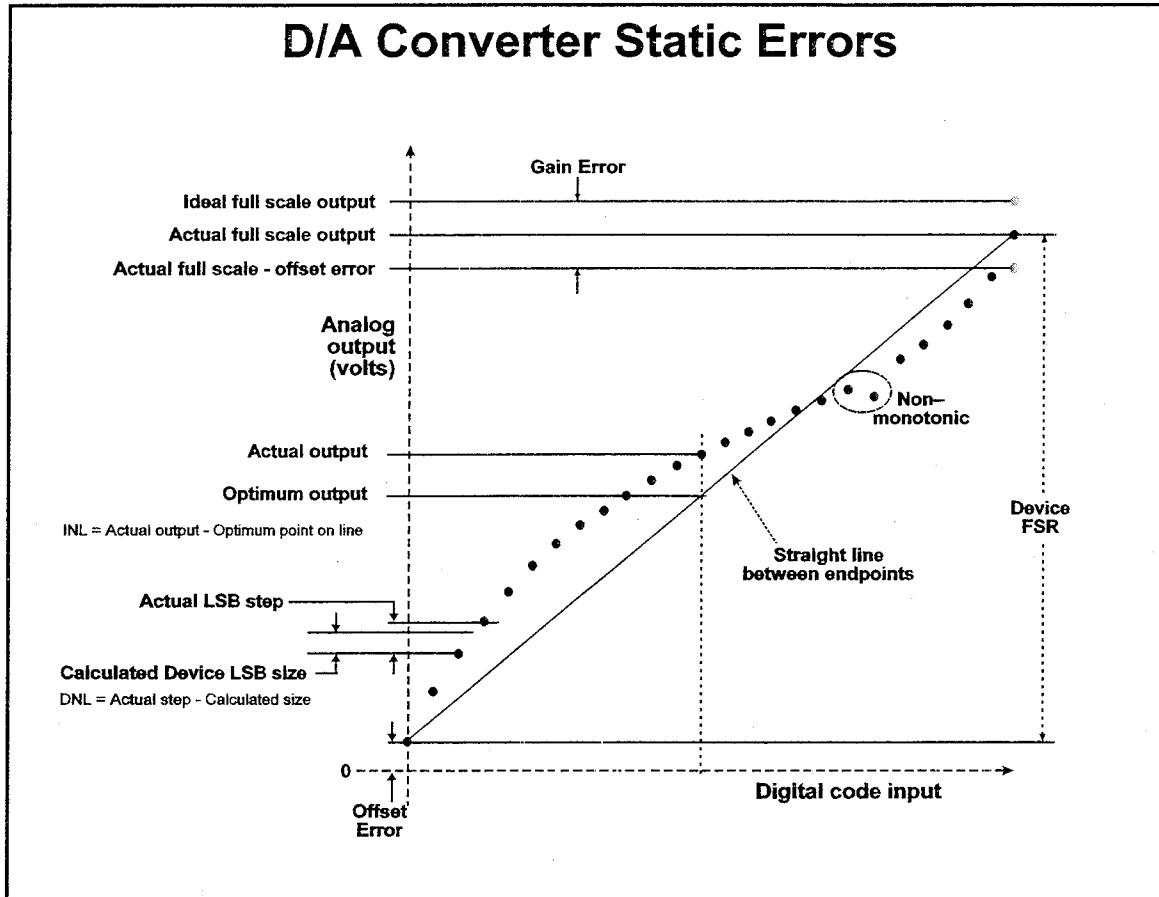


Figure 3.1

NOTES:

LSB size

A linear DAC transfer characteristic is shown as a series of points along a 45 degree line. The binary input codes shown on the left of the figure increment by 1 for each successive code, counting from, for example, 0 to 255 for an 8 bit DAC. The output should in turn increment by

$$FSR \times \frac{1}{2^{bits} - 1} \quad (3.1)$$

where FSR is the full scale output range of the DAC. The definition of one *least significant bit* or LSB is the expected output change due to a change of input code by the minimum amount. An ideal LSB is calculated from the specified FSR whereas when testing, an LSB is an expected average value based on the actual length of the transfer curve. In other words it depends on each specific device's offset and gain errors.

Differential nonlinearity (DNL)

The DAC transfer characteristic found most often is a *linear DAC*, in which the analog output signal has a 1:1 correlation to a binary digital input code. (Other transfer characteristics are, for example, logarithmic output and "rotational" output of degrees.) If a line is drawn between the first and last point of a linear DAC transfer graph, any point which deviates from that line is considered a linearity error. Differential nonlinearity is a measure of the "small-signal" linearity error, and is defined as the difference in the output voltage at a specific input as compared to the output at the previous input plus 1 device LSB.

Monotonic

A word to state that, with increasing input code, the output stays the same or increases and *vice versa*. This quality of a DAC is very important if it is used in a feedback loop. When a non-monotonic device is used in a feedback loop, the loop can get "stuck" such that the DAC toggles (or oscillates) forever between 2 input codes. Monotonicity is guaranteed if DNL is less than ± 1 LSB. It is often specified as a number of bits which is equal to or less than the resolution of the device, e.g. a 14 bit DAC may be specified as "monotonic to 12 bits."

Integral nonlinearity (INL)

If differential nonlinearity is the small-signal linearity error, integral nonlinearity is the large-signal nonlinearity error. Integral nonlinearity is the cumulative effect, for any given input, of all differential nonlinearities. It may also include other sources of nonlinearity, a classic case being that of DAC output voltage changing because of resistive heating effects as DAC currents change in the R/2R ladder. Integral nonlinearity is measured by testing how far a given output deviates from a line drawn between the transfer function end points.

Accuracy

How well a DAC matches a perfect device. This includes the errors noted above and may be given in percent of reading similar to the way voltmeters are specified. It is calculated from static errors, not measured directly.

NOTES:

Other parameters of interest

Maximum conversion rate

This parameter is somewhat self explanatory. As a DAC's input changes, its output must be given time to reach the output level and settle. The value of this parameter should include the worst case rate, which is most likely the inverse of the time required to change from zero scale to full scale output.

Settling time

Essentially the same as op amp settling time. It is the time required for the output to reach and remain within $\pm\frac{1}{2}$ LSB of its final value or within some other defined limit. It may be expressed in LSBs or in %FSR, for example $\pm 0.01\%$.

NOTES:

Test System Configuration for DAC Static Parameter Tests

As you can see in Figure 3.2, the WD, WG and DSP components of a mixed signal test system are not required for static linearity for a DAC. These measurements are done more in the manner of a PMU test for a digital pin, although the accuracy requirements are much more stringent. Depending on the DAC specifications, the *High Accuracy Measure Unit* could be the tester's WD.

The digital subsection is required to drive the DAC input bits and other digital pins such as write (WR), chip select (CS) or clock (for serial input DACs). DAC timing requirements must be met.

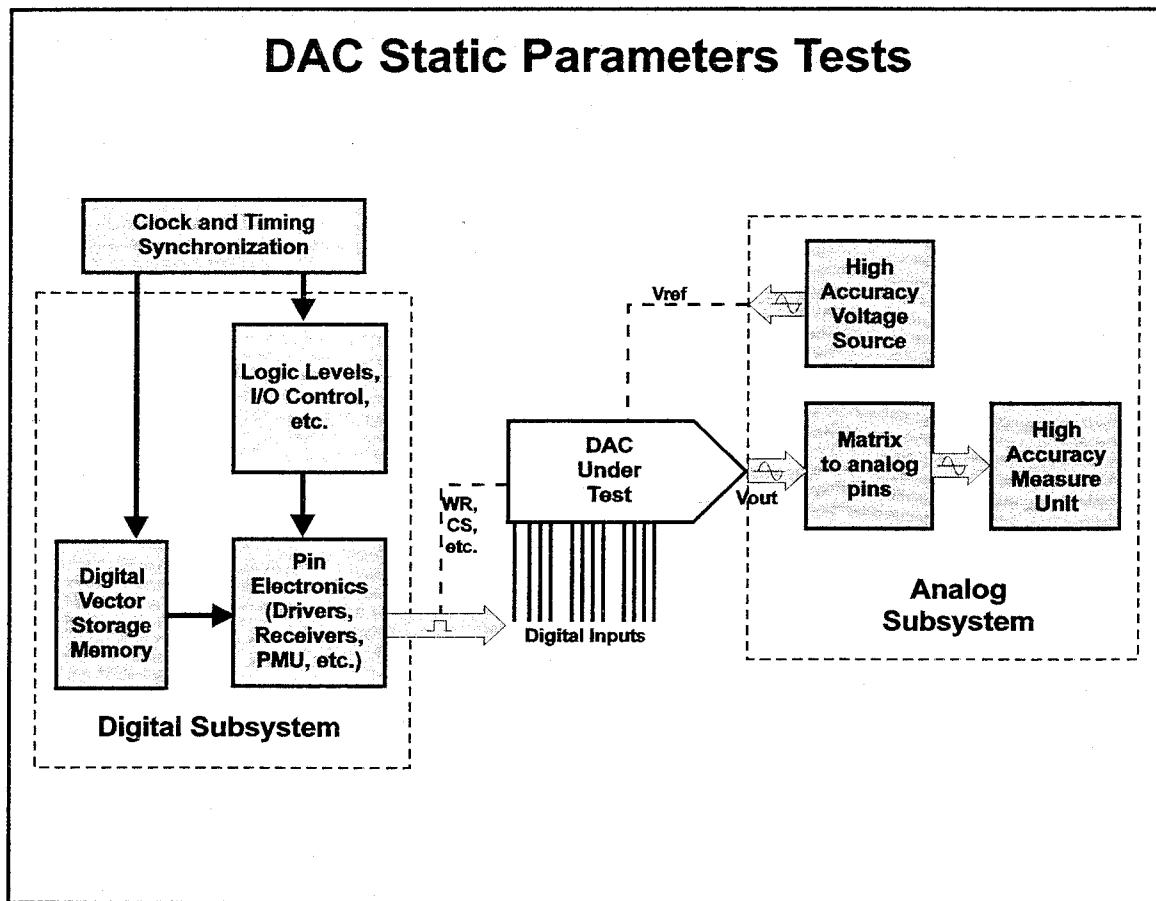


Figure 3.2

NOTES:



Example DAC data sheet

The following table is an example of a DAC data sheet specification; it clearly shows which parameters are static—offset and gain error, differential and integral nonlinearity and output range.

Note that the limit values for offset and gain are specified in %FSR, which stands for *percent full scale range*; DNL and INL are specified in LSBs. Both of these units are “DUT- relative” not absolute. We will discuss how to convert measurable units such as volts and amps into percent full scale range and LSBs.

Parameter	Conditions	Min	Typ	Max	Units
INPUT					
Resolution		12			Bits
Input format	straight binary				
STATIC					
Offset error	input = 0x0			±0.2	% FSR
Gain error	input = 0xFFFF			±0.4	% FSR
Differential nonlinearity	guaranteed monotonic to 12 bits			±1	LSB
Integral nonlinearity				±½	LSB
Output range			±2.5		V
DYNAMIC					
SNR	fout = 1000Hz sinusoid	68	70		dB
THD	fout = 1000Hz sinusoid, fmax=20KHz		-69	-67	dB
IM	fout = 1000Hz + 7000Hz tone			-65	dB
AC					
t _{settle}	Max input change = 16 LSBs		180	200	nsec
Conversion rate				5	MHz

Table 3.1 DAC Specifications

NOTES:

Parameter Measurement Requirements

As noted above, the static parameters are specified relative to the device under test. In other words, they are not measured directly as a voltage or current, but require multiple input stimuli and output measurements. These measurements are then used to calculate the parameters that are compared to the specification limits.

Offset and gain error calculations require zero and full scale measurements, which are also needed for DNL and INL measurements. Thus offset and gain are discussed first. (The *Resolution*, *Input format* and *Output range* specifications seen in Table 3.1 give information required for testing and are not true test parameter limits.)

Parameter to test	Measurement(s) required	Items needed for measurement(s)
Offset error	zero scale output value	◊ zero scale input code
Gain error	1. zero scale output value 2. full scale output value	◊ zero scale input code ◊ full scale input code
DNL	1. zero scale output value 2. full scale output value 3. output values for pairs of adjacent input codes	◊ DAC resolution in bits ◊ FSR ◊ DUT LSB size ◊ input codes to test
INL	1. zero scale output value 2. full scale output value 3. output values for selected input codes	◊ DAC resolution in bits ◊ FSR ◊ Equation for line between zero scale and full scale ◊ input codes to test

Table 3.2 Measurement requirements for static parameter testing

NOTES:

Measuring offset and gain errors

Why do we need to measure offset and gain errors? For two reasons—first because it is in the specification and is an important parameter for customers who use DACs. Second because both offset and gain errors affect the output measurements for all input codes, so the offset and gain errors (in the form of zero and full scale measurements) must be factored into our linearity measurement calculations later.

As can be seen in Table 3.2 on page 66, offset, gain, DNL and INL measurements all require a zero scale measurement. As a result, offset error is the starting point for all static measurements. Gain error, DNL and INL require zero and full scale measurements, so they follow in order.

Zero scale measurement conditions

A DAC has a range of digital input values and corresponding analog output values. Depending on the data format of the DAC, zero output may occur when the input is at all zeros (straight binary), all ones (complement binary) or at MSB = 1 and all other bits = 0 (two's complement binary or sign-magnitude binary). Usually, the two input codes at either end of the range correspond to the zero and full scale output values. With a straight binary input device as considered here, all zeros in gives zero scale out and all ones in gives full scale out.

For a perfect DAC, the zero scale input code produces zero scale output and the full scale input code produces full scale output. For example, a voltage output DAC with a 0 to 5V output range has zero scale out = 0V and full scale out = 5V. *Offset error* and *gain error* are the difference between the ideal output values and the actual DAC output values at zero and full scale respectively.

NOTES:

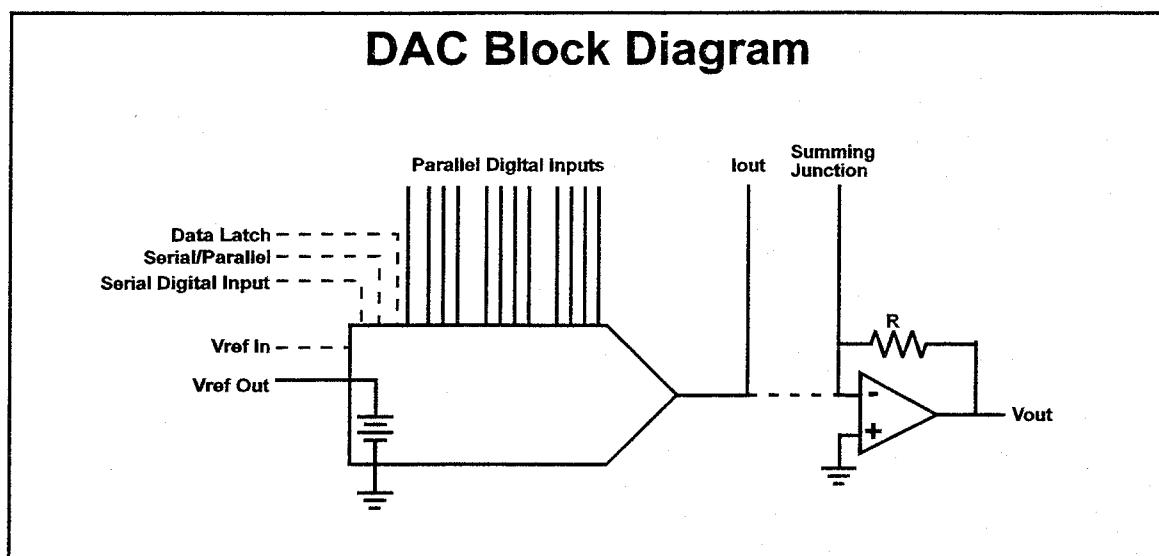


Figure 3.3

In Figure 3.3, if the *Summing Junction* and *I_{out}* pins are connected, zero scale *V_{out}* exists when no current flows through $R\Sigma$; full scale *V_{out}* exists when the DAC's maximum current flows through $R\Sigma$. For zero scale out, any leakage current from the current DAC or the op amp input and any input offset voltage of the op amp will make the zero scale output value not equal zero.

For full scale out, any output current error from *I_{out}* or in *V_{ref}* will cause an error in the maximum output voltage. Also, notice that at full scale out, the op amp offset voltage and other zero scale error signals are still present. That means that *the error for zero scale output (the offset error), also affects full scale output and all output values in between*. Thus a full scale output measurement contains an offset error component and a gain error component.

NOTES:



Offset Error

Offset error is similar in nature to V_{IO} for an op amp. In the case of a current output DAC, offset error is the output current when no current should flow from the device, such as leakage current. For our example of a unipolar straight binary voltage output DAC, the output voltage with all zeros at the input should be 0V. any deviation from 0V is offset error. It is the value measured at zero scale minus the ideal (specified) value; mathematically (for a voltage out DAC):

$$\text{Offset Error} = V_{ZS} - V_{ZS[\text{ideal}]} \quad (3.2)$$

Translating limits to something you can measure

Offset error can be specified in volts, current (for current out DACs), LSBs, percent (parts per hundred) or PPM (parts per million) of full scale. In all cases we must convert the specification to what we are measuring, e.g. volts or amps, to make a limit comparison in our test.

Question 3.1: Assume you are testing a DAC with an offset error specification of $\pm 0.2\%$ FSR and a $\pm 2.5\text{V}$ output range. What is the measurement limit in volts to be used in a test program?

Question 3.2: Assume you are testing a unipolar DAC (digital zero in gives 0V out ideally) with an offset error specification of $\pm 800\text{ppm}$ FSR and FSR = 10V. What is the measurement limit, in volts, for a test program?

Question 3.3: Assume you are testing a unipolar DAC with an offset error specification of $\pm 100\text{LSBs}$ on a 14 bit DAC with 5V FSR. What is the measurement limit in volts to be used in a test program?

NOTES:

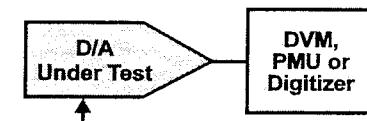
Making the zero scale measurement

Now that we can calculate a useful measurement limit, how do we make the measurement? Offset error measurement is an *absolute* measurement because it is compared with an absolute limit value (versus a *relative* measurement where the error is relative to another measured value). Often the standard PMU on a test system will not have sufficient accuracy to measure offset error. If the specification limit is $\pm 10\text{mV}$, the instrument making the measurement should be at least 10 times more accurate than the expected measurement value. Whatever makes the measurement should then have an absolute accuracy of better than $\pm 1.0\text{mV}$; often this requires the use of an integrating DVM type of instrument rather than a tester PMU. This is illustrated in Figure 3.4 as "Direct Measurement."

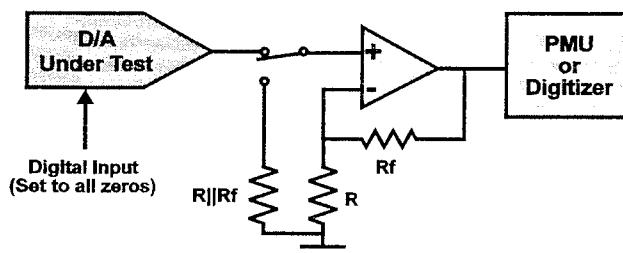
Instead of a high accuracy meter (which tends to be somewhat slow), another way to measure offset is to amplify it to a larger value so it can be measured with a PMU. Recognize that amplification can introduce its own offset error so it must be carefully designed and applied. The switch in the "Offset Amplification" diagram of Figure 3.4 is used to measure the offset of the amplifier alone so it can be subtracted from the final measurement.

NOTES:

D/A Offset Measurement



Direct Measurement



Offset Amplification

Figure 3.4

Question 3.4: For the 12 bit DAC specified in Table 3.1 on page 65, a zero scale measurement of -2.50134V is taken. Using the Output Range given in the same table, what is the offset error (in % ideal FSR)?

NOTES:

Gain Error

Gain error is the difference in the specified ideal value and the actual output value when the input digital value is set to full scale. For our unipolar straight binary DAC, all ones represents full scale input. For a current output DAC, maximum current flows from the device; for voltage output, the analog signal is the maximum voltage the DAC can supply.

Graphically, gain error is the error in the slope of the line between the zero scale and full scale values. It is specified as the deviation in output from the ideal value *after compensating the measurement for offset error*. By drawing the line from zero scale to full scale (rather than from the ideal zero point to full scale) offset error is automatically compensated. As with offset error, gain error can be specified in units of per cent FSR, ppm FSR, LSBs or in the output units such as millivolts or microamps.

In Figure 3.1 on page 61, note the point which represents “offset plus ideal FSR” and that it will be shifted down when offset error is subtracted from it. The measured full scale value is less than the offset compensated ideal full scale output. In practice, gain error is calculated as:

$$\text{Gain Error} = (V_{FS} - V_{ZS}) - V_{FSR[\text{ideal}]} \quad (3.3)$$

Question 3.5: For the 12 bit DAC specified in Table 3.1 on page 65, a zero scale measurement of -2.50134V and a full scale measurement of +2.48874V are taken. What is the actual FSR for this device?

Question 3.6: Using the Output Range given in Table 3.1 on page 65, what is the gain error (in %ideal FSR)?

NOTES:

Question 3.7: What is the effect of the non-monotonic code shown in Figure 3.1 on the overall gain (i.e. full scale output) of the device? (Recall that the full scale output is the cumulative sum of all individual bit values.)

Question 3.8: Does this device pass the offset and gain specifications?

NOTES:

Measuring DNL and INL

Differential and integral non-linearity are two traditional static measurements for the output quality of a DAC. "Static" means the input is set and a DC measurement of the output level is made. Conceptually, DNL and INL are simple; in practice there are (as usual) trade-offs to make regarding test coverage and test time.

Making one DNL measurement

A differential linearity check requires four measurements. The first two are zero and full scale, which were already made during the offset and gain error tests. They are used to calculate the device LSB size as given by:

$$LSB = \frac{FSR_{device}}{2^{bits} - 1} \quad (3.4)$$

Question 3.9: For the 12 bit DAC specified in Table 3.1 on page 65, a zero scale measurement of -2.50134V and a full scale measurement of +2.48874V are taken. What is the LSB size for the device?

Did you get LSB size = 1.219mV? With the LSB size calculated, we can now measure the DNL at any point on the DAC transfer curve. Recall the DAC static error diagram in Figure 3.1 on page 61. The ideal step in this case is the calculated LSB size and an actual measured LSB step requires setting the DAC to 2 adjacent codes one at a time and measuring the output voltage.

Suppose we want to measure the DNL at the major transition code at mid-scale. That is the step size between when the MSB is low and all other bits are high and when the MSB is high and all other bits are low. The DAC from Question 3.9 has measurements made for the 2 input codes at its mid-scale as shown in Table 3.3:

Code	Output Voltage
011111111111	-0.1136127 V
100000000000	-0.1122525 V

Table 3.3

NOTES:



Question 3.10: What is the result (in volts) of measuring the 2 points and subtracting the first from the second?

Question 3.11: What is the DNL at the major transition in V? (Actual - LSB size) [Hint: you need the correct answer from Question 3.9]

Question 3.12: What is the DNL as a fraction of LSB size? [(Actual - LSB size) / LSB size]

Question 3.13: Is the DNL at this transition within specification?

Question 3.14: Why are so many significant digits used in these calculations?

Making all DNL measurements

Important input codes

The question becomes “Which codes must I test?” For converters with a resolution of 8 bits or less (maximum 256 measurements), testing all codes normally does not make for an overly long test time (unless the meter used to make the measurements is extremely slow). With 12 bit resolution and higher devices, there are many codes to test. The task becomes finding the fastest way to measure the least number of codes which yield the necessary information for linearity error calculations.

Generally, which codes are tested depends on the architecture of the DAC. For a standard R/2R architecture, testing the major transitions is sufficient. If there is a possibility of *superposition error*, (see *What is Superposition Error?* on page 78) it is also necessary to test the one’s complement counterparts of the major transitions. This is

NOTES:

Digital to Analog Converter Static Measurements

the best case scenario, in which the number of transitions to test equals approximately twice the number of bits.

Input Code (decimal)	Major Code Transitions for 12 bit DAC
	MSB.....LSB->MSB.....LSB
0 -> 1	000000000000->000000000001
1 -> 2	000000000001->000000000010
3 -> 4	000000000011->000000000100
7 -> 8	000000000111->000000001000
15 -> 16	000000011111->000000010000
31 -> 32	000000011111->000000100000
63 -> 64	000000111111->000001000000
127 -> 128	000001111111->000010000000
255 -> 256	000011111111->000100000000
511 -> 512	001111111111->001000000000
1023 -> 1024	001111111111->010000000000
2047 -> 2048	011111111111->100000000000

Table 3.4

There are R/2R DACs which have some of the upper bits *decoded*. Decoding puts the larger currents through more resistors, reducing temperature drift and making it easier to accurately adjust the upper bits during manufacture.¹

If the upper 2 bits are decoded, 2 bits become $4 (2^2)$ switches and resistors; if the upper 3 bits are decoded, 3 bits become $8 (2^3)$ switches and resistors, etc. If you've noticed, the requirement for how many transitions to measure depends on how many current paths are in the DAC. With decoding, the number of current paths is increased by $(2^{\# \text{bits decoded}} - 1)$, so how many additional transitions must be tested? See Figure 3.5 for a schematic of a 12 bit DAC with its upper 2 bits decoded.

NOTES:

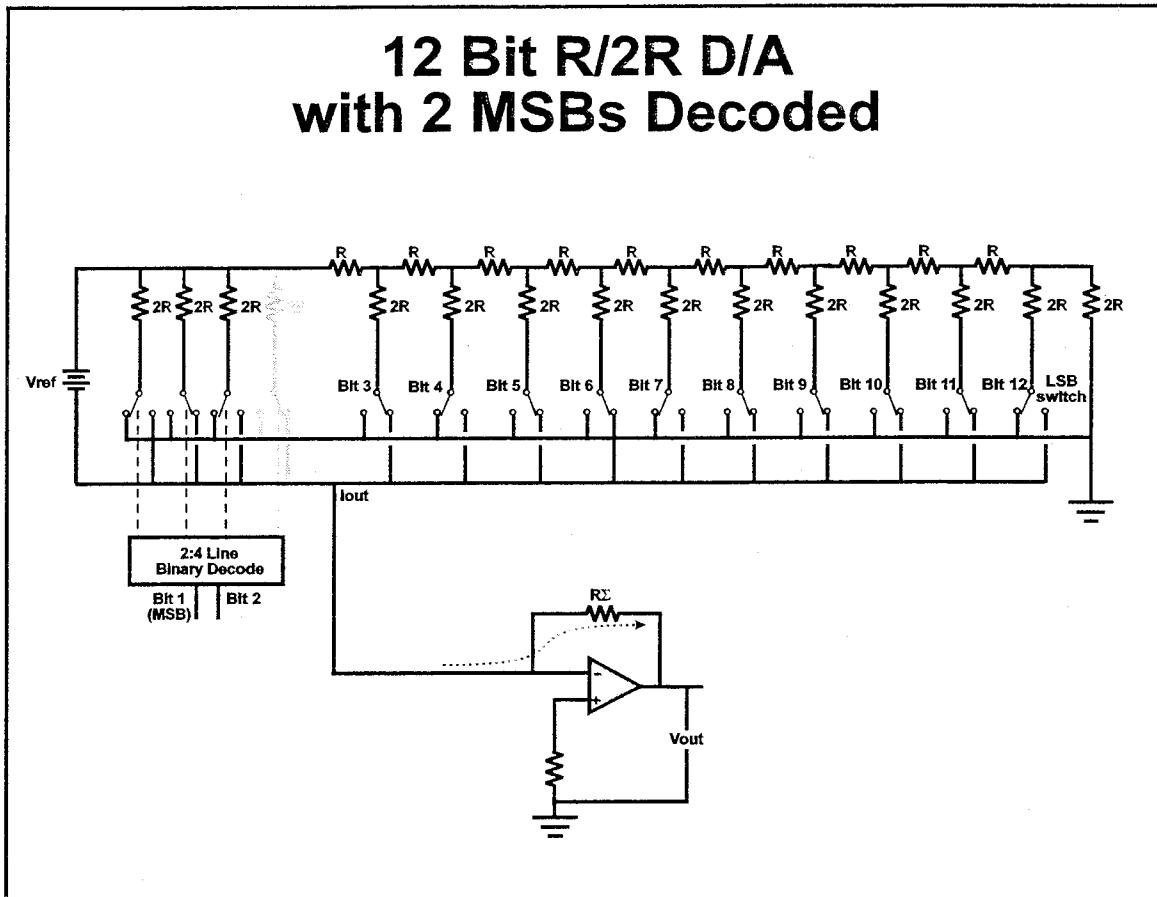


Figure 3.5

The additional input codes required for this DAC are shown below, with decoded bits separated from the rest:

Input Code (decimal)	Transitions for 2 decoded bits in 12 bit DAC
	MSB.....LSB->MSB.....LSB
511->512	00 1111111111->01 0000000000
2047->2048	01 1111111111->10 0000000000
3071->3072	10 1111111111->11 0000000000

Table 3.5

NOTES:



The grayed switch in Figure 3.6 illustrates that only 3 of the 4 decoded states require resistors; when bits 1 and 2 are 0, no current is added to I_{out} thus no resistor is needed for that state. Notice that the only additional codes not already included in the 12 codes given for the major transitions are 3071 and 3072. The performance improvement is more noticeable and the additional testing more obvious when more bits are decoded, for example the upper 4 bits of a 16 bit DAC.

Check with the design department to learn about the architecture of the converter and to find the relevant codes to test. If the device was manufactured elsewhere, you may be able to discern its architecture from the data sheet or you may need to contact the manufacturer.

When you don't know the architecture or if you know you are testing a video DAC, you must test all codes. This is not really as difficult as it sounds—if you can make a single measurement in 10msec, DNL for a 16 bit DAC (65536 codes) with superposition testing will take about 650msec. To achieve a measurement with the necessary accuracy, a direct measurement may require a slow integrating DVM controlled by a slow IEEE-488 or RS-232 interface, extending the test time substantially.

Question 3.15: A 14 bit device (not decoded) must be tested for DNL at major transitions. How many measurements must be made?

What is Superposition Error?

Superposition is, as you probably recall, when the whole equals the sum of all its parts. The classic example is when you can separately calculate all currents in multiple circuit network loops and add them together to get the total current in a wire common to those loops.

In a DAC, superposition should hold. Setting a specific input bit should produce the same component voltage drop across the summing output resistor no matter which other bits are on or off. For example, suppose an input code of 000000000001 is set at the DAC input and the output voltage is 2.44mV; set an input code of 100000000000 and the output voltage is 5.00000V. If you then set 100000000001 as the input code, you expect to get 5.00244V as the output. If you do not, this is a superposition error. DACs with no superposition error have a transfer graph which is a mirror image across the horizontal axis between zero to mid-scale and mid-scale to full scale.

The primary cause of superposition error is self-heating of the output summing resistor from the power it must dissipate when current flows through it, causing a code dependent resistance value.² If the summing resistor does not change with temperature (*temperature coefficient of resistance*, or $TCR = 0$), there is no superposition error in the summing resistor.

This type of superposition error occurs because the thin film resistors used in integrated DAC circuits cannot dissipate very much heat. If an external resistor is used instead, the error can be eliminated. However, an external summing resistor does not have the same TCR as those in the resistor ladder in the DAC circuit, introducing an entirely different temperature related problem.

NOTES:



Faster Measurements

Fortunately, there are ways to make DAC output measurements faster; as always, the best technique depends on the DUT. For low resolution DACs, a digitizer or PMU with high resolution can make direct measurements. Low resolution DACs are often tested with the DUT input simply counting from zero scale to full scale and the output measured directly with a digitizer or PMU. Higher resolution devices require a more sophisticated approach.

Video DAC

Video DACs are normally low resolution devices, on the order of 8 bits. A 12 to 16 bit digitizer or accurate PMU can adequately measure the output values. The data to the video DAC must be synchronized with the measurement device, whether it is a digitizer or a PMU. The technique is simple—change the DAC input, wait for the output to settle then digitize the output value. It is likely that the DAC will be faster than the digitizer; when this is true, the time between measurements is limited by the time between samples and not the DAC settling time. The DAC will thus not be tested at its maximum conversion rate.

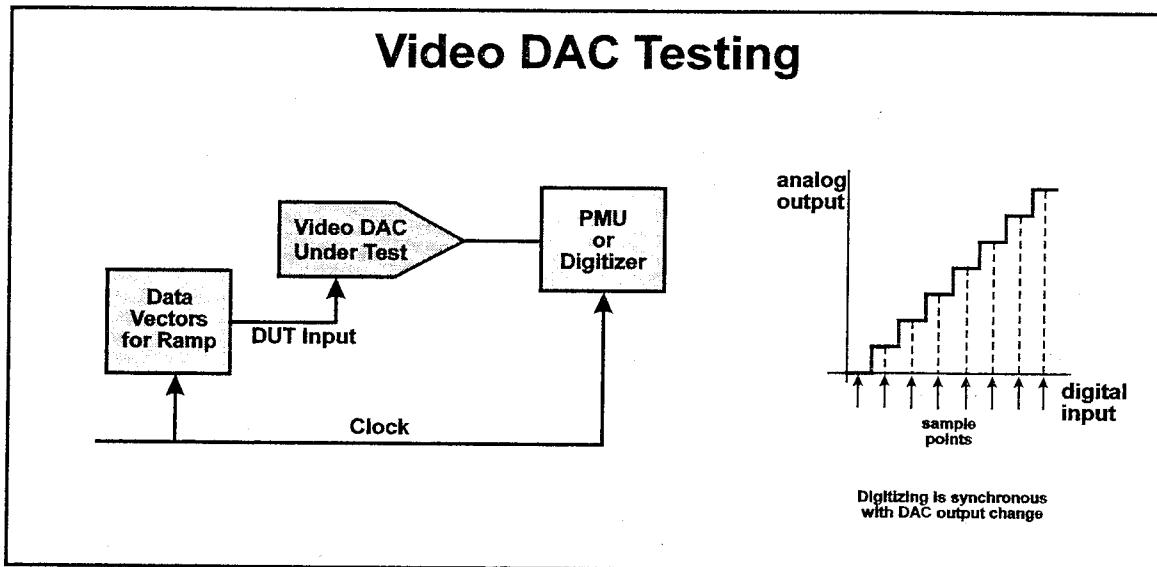


Figure 3.6

NOTES:

High Resolution DACs

When the DAC under test has a high resolution or small LSB size (<5mV), it may be necessary to amplify the output error to rapidly measure the DUT output. Figure 3.7 shows a calibrated reference DAC which is used as a "perfect signal." By subtracting the DUT output signal from the reference signal, an amplified error signal is created which can be measured with a faster and more local instrument such as a PMU or a waveform digitizer. The gain factor for the difference amplifier is dependent on the quality of the measurement device.

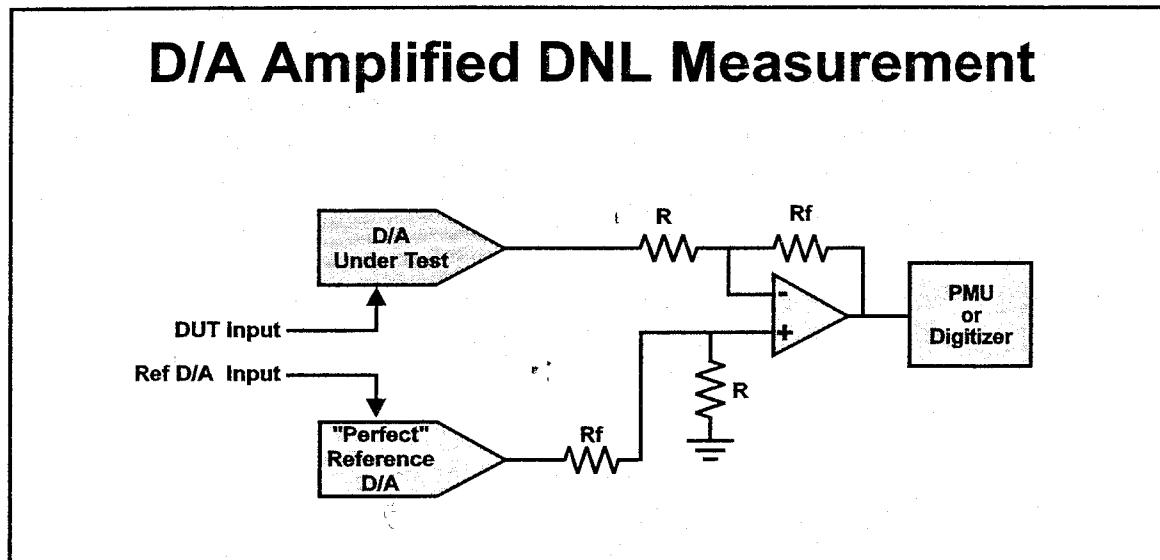
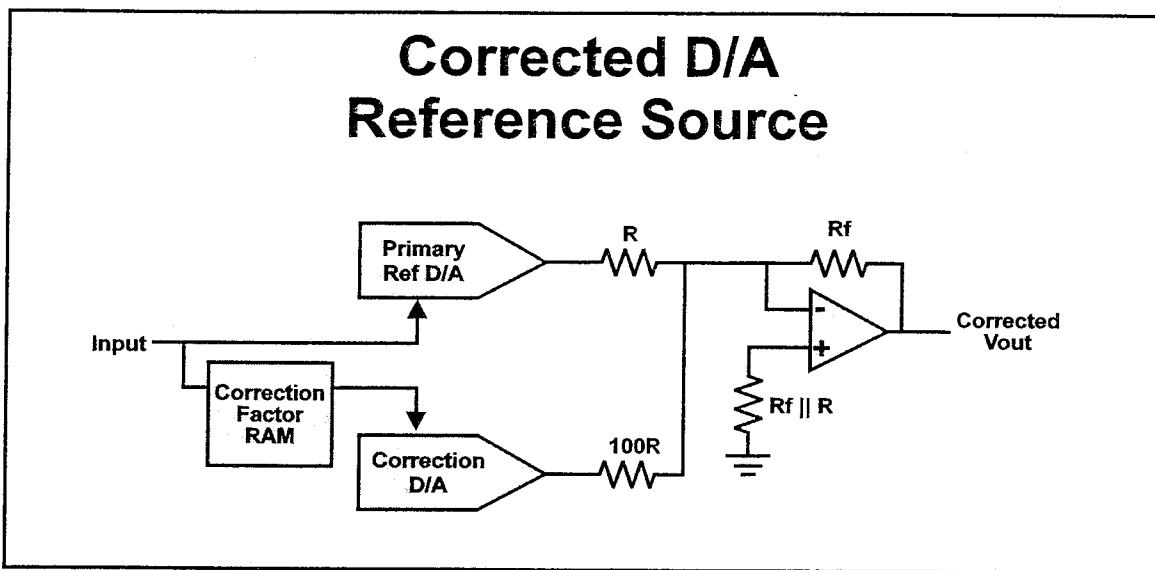


Figure 3.7

The degree of "perfection" for the ref DAC depends on the DUT. If the DUT is 14 or 16 bits or 12 bits with a small output range (i.e. if the DUT has an LSB size below 5 mV), a calibrated ref DAC may be required. Some mixed signal test systems have a highly accurate and calibrated reference source, perhaps in a temperature controlled chamber. You can also use an external high accuracy voltage source, but this may also have a slow interface. As a last resort, you can design your own high accuracy reference source, using 2 DACs and some RAM as shown in Figure 3.8. (The similarity between Figure 3.7 and Figure 3.8 can be confusing; Figure 3.8 fits entirely within the *Perfect Reference D/A* shown in Figure 3.7.)

The RAM serves as a lookup table which, for a given input code on the Primary Ref DAC, sends code to the Correction DAC which corrects any error of the Primary Ref DAC at that code. Thus the RAM must be big enough to hold a value for each code of the DAC, e.g. 4K for a 12 bit DAC or 64K for a 16 bit DAC (K represents 1024 in this context, not 1000).

NOTES:

**Figure 3.8**

Notice that the output of the Correction DAC is attenuated by a factor of 100 compared to the Primary Ref DAC output. This decreases the output range but increases the voltage resolution, so very fine adjustments are possible. It also, coincidentally, attenuates any noise from the Correction DAC by a factor of 100. Correction factors must be stored during test system calibration and loaded before testing begins.

NOTES:

The DNL Test

Once you have a set of DNL error values, how do you test for DNL? The test requires comparing each error value to the specification. If they are all within the specification, the device passes DNL. As soon as a value is outside the ± 1 LSB limit, stop testing and reject the device or downgrade it and continue. You can save test time by doing limit comparisons when measurements are made, rejecting bad devices before measuring all codes.

Question 3.16: What is monotonicity? [Hint: see the section *Monotonic* on page 62]

Question 3.17: Why does a DNL limit of ± 1 LSB (2 LSBs total) guarantee monotonicity?

Question 3.18: On an R/2R DAC, would you test codes above mid-scale for DNL?

Question 3.19: How many codes should you test on an 8 bit video DAC?

NOTES:



The INL Test

As you can see, measuring DNL by measuring the output values of pairs of DAC input codes is fairly simple. After testing DNL, we should have all the data necessary to test INL; only value comparisons are necessary. The code value which is furthest from the straight line between the zero scale and full scale endpoints is the maximum INL. If the difference between the expected point on the line and the measured value at that code is within INL limits, the DUT passes; if not, it fails. Make sure code values for superposition error are tested.

Using the equation for a straight line, $y = mx + b$, allows a calculation of each output point value. The slope m is the change in output divided by the change in input, the y intercept b is the offset measurement value, x is the input code and y is the output value. In equation form:

$$\text{Output} = \frac{\text{FullScale} - \text{ZeroScale}}{2^{\text{bits}} - 1} (\text{InputCode}) + \text{OffsetError} \quad (3.5)$$

This can be rearranged to set a ratio of InputCode to 2^{bits} to reduce calculation errors as follows:

$$\text{Output} = (\text{FullScale} - \text{ZeroScale}) \left(\frac{\text{InputCode}}{2^{\text{bits}} - 1} \right) + \text{OffsetError} \quad (3.6)$$

As a software algorithm, INL can be tested in a loop as shown, assuming an array of input codes has been created and an array of corresponding output measurements has been taken and stored:

```

ZeroScale = OutputMeasured[0];
FullScale = OutputMeasured[CodeCount - 1];
OffsetError = ZeroScale - IdealZeroScale;
FSR = FullScale - ZeroScale;
NumBits = pow(NumberofBits, 2);
LSB_Volts = FSR / NumBits - 1;
for (Index = 0; Index < CodeCount; Index++) {
    ExpectedOutput = FSR * (InputCode[Index] / (NumBits-1)) + OffsetError;
    INL_Volts[Index] = OutputMeasured[Index] - ExpectedOutput;
    INL_LSB[Index] = INL_Volts[Index] / LSB_Volts;
    if abs(INL_LSB[Index]) > 0.5 {
        FailDevice(InputCode[Index], OutputMeasured[Value]);
        break;
    }
}

```

Question 3.20: On an R/2R DAC, why would you test codes above mid-scale for INL?

NOTES:

Current Output DAC

If you are testing a voltage output DAC, everything in this chapter is directly applicable. If you are testing a current output DAC, you will need to convert the current to a voltage with the correct full scale and polarity for the PMU or waveform digitizer on your test system as shown in Figure 3.3 on page 68. Here are some guidelines for converting current output to voltage output:

1. Use the DAC's internal resistor for the summing resistor. It is manufactured to match the other components in the DAC with temperature and value.
2. If you take measurements with a WD and the full scale or polarity of the converted output does not match the WD's input range, use an attenuator or gain stage to adjust the full scale; use a unity gain inverter to reverse polarity.
3. Add a relay or two to allow a direct measurement of the amplifier (or attenuator) circuit with its input shorted to DUT ground; subtract this measured value from each measurement of the DAC. This is an auto-null technique which compensates for circuitry external to the DUT. This is illustrated in Figure 3.4 on page 71 which shows DAC offset measurement.

NOTES:

Key Points of This Chapter

- ❖ Offset, gain, DNL and INL are the primary static specifications for DACs.
- ❖ Offset and gain are absolute measurements; DNL and INL measurements are referenced to the zero and full scale outputs of the DAC.
- ❖ DNL requires 2 measurements for each error calculation, in addition to zero and full scale measurements to calculate device LSB size.
- ❖ INL can be calculated from measurements made for the DNL test.
- ❖ Test time can be optimized by knowing the minimum number of input codes required to characterize DAC linearity. The minimum number of required input codes depends on the architecture of the DAC.
- ❖ DAC error signals can be very small, especially for low output range devices and high resolution devices. It may be necessary to use error amplification techniques to make the measurements quickly with sufficient accuracy.

NOTES:

Answers to chapter questions

Answer 3.1: The specification is 0.2% FSR; what is the measurement limit? We must know the FSR to figure it, so we find the FSR specification is 5V. First convert from percent (per 100) to "per 1" by dividing by 100. This gives $V_{os} = 0.002$ and $FSR = 0.002 \times 5V = 10mV$. Thus the limit is $\pm 10mV$ above and below the ideal zero scale value (-2.5V for our DUT).

Answer 3.2: Convert from ppm to "per 1" by dividing by 1 million: $\frac{800}{10^6} = 0.0008 \times FSR$. Multiply by FSR = 10V and the limit is $\pm 8mV$.

Answer 3.3: One LSB = $\frac{5}{2^{14}-1} = \frac{5}{16383} = 3.052 \times 10^{-4}$. Multiply this by 100 LSBs to get $\pm 30.52mV$ limits.

Answer 3.4: $-2.50134V - (-2.5V) = -1.34mV$ offset error.

Converting to %FSR, $(-1.34 \times 10^{-3}V / 5V) * 100\% = -0.0268\%FSR$

Answer 3.5: $FSR = 2.48874V - (-2.50134V) = 4.99008V$

Answer 3.6: Gain error = $FSR - \text{Ideal FSR} = 4.99008 - (2.5 - (-2.5)) = 4.99008 - 5 = -9.92mV$

Converting to %FSR, $(-9.92 \times 10^{-3} / 5) * 100\% = -0.1984\%FSR$

Answer 3.7: It decreases overall gain.

Answer 3.8: Yes

Answer 3.9: LSB size = $FSR / 4095 = 4.99008V / 4095 = 1.219mV$

Answer 3.10: Step size = $1.3602mV$

Answer 3.11: $1.3602mV - 1.2186mV = 0.1416mV$

Answer 3.12: $0.1416 / 1.2186 = 0.116\text{LSB}$

Answer 3.13: Yes.

Answer 3.14: Because we are dealing with such small quantities compared to FSR, and 0.1LSB is a measurement accuracy goal. In other words, we would like to be confident that we can measure DNL and INL at 1/10 LSB accuracy. Rounding can cause cumulative errors in calculations that would prevent that level of accuracy, even with perfect measurement values.

Answer 3.15: DNL requires 27 measurements for 14 code widths up to mid-scale (not 28, because the code at 1 can be used for DNL at 0 and 2). Full scale must also be measured, so the final count = 28.

NOTES:

Answer 3.16: The characteristic that, when the input changes in a certain direction, the output always changes in a related direction; conversely, an output change can never have 2 different signs for a same-signed input change.

Answer 3.17: If an input change of one LSB causes no change at all on the output, the output has not violated the "no change of direction" characteristic. An output change $> -1\text{LSB}$ is a non-monotonic event.

Answer 3.18: Only if the DAC has decoded bits.

Answer 3.19: a) 256

Answer 3.20: To test for superposition error.

NOTES:

References

1. Sam Wilensky, Sipex Corp., E-mail correspondence, June 29, 1998
2. *The Handbook of Linear IC Applications*, pg 41-46, 57-61. Burr-Brown Corp. 1987

NOTES:

Analog to Digital Converter Static Measurements

Goals

- ❖ Study ADC static measurement specifications
- ❖ Discuss the unique difficulties in ADC testing
- ❖ Examine various circuit techniques to overcome these difficulties
- ❖ Understand measurement techniques for offset and gain error
- ❖ Examine ADC output data capture
- ❖ Expose various measurement techniques for DNL and INL parameters--histograms, transition measurement, inferral by Effective Number Of Bits (ENOB)

Objectives

- ❖ Understand how to measure a specific point on an ADC input-output transfer line
- ❖ Know how to measure and calculate ADC zero and full scale values
- ❖ Be able to connect the digital pins of an ADC for continuous conversion and retrieve the digital output data into a test system
- ❖ Recognize the inherent noise in ADC measurements
- ❖ Know of various analog interface circuit options to find ADC transition points
- ❖ Be able to measure points on an ADC input-output transfer line and use them to calculate differential and integral non-linearity (DNL and INL)
- ❖ Understand histogram testing of ADCs
- ❖ Know the trade-offs of different DNL/INL test techniques

What you will learn

- ❖ ADC static specification parameters—what they represent and how they relate to measured values
- ❖ How measuring points on an ADC transfer line is “backwards”
- ❖ How to interface with the digital pins of an ADC so it will continuously convert
- ❖ How to do a histogram test for DNL and INL
- ❖ A closed loop system to allow measurement of ADC transition points
- ❖ Ways to make ADC testing faster
- ❖ How to find DNL and INL with static measurements
- ❖ How to calculate LSB size, offset error, gain error, DNL and INL

NOTES:

This chapter covers the static measurements of an ADC. It includes how to measure the static specification parameters of an ADC. It also covers how to interface with the digital pins of an ADC so it will continuously convert. It includes how to do a histogram test for DNL and INL. It also covers a closed loop system to allow measurement of ADC transition points. It includes ways to make ADC testing faster. It also covers how to find DNL and INL with static measurements. Finally, it covers how to calculate LSB size, offset error, gain error, DNL and INL.



Overview of Testing ADC Static Parameters

Before discussing how to test static ADC parameters, let's review these things:

- ◊ ADC specifications
- ◊ A general test system configuration for ADC static parameter testing
- ◊ An example ADC data sheet
- ◊ A diagram of the errors
- ◊ A list of ADC static parameters and the measurements required to calculate their values

NOTES:

ADC Static Specifications

Static ADC specifications describe how well the ADC conforms to the input versus output transfer curve for which it was designed. They are:

- ❖ Full scale range (FSR)
- ❖ Offset error
- ❖ Gain error
- ❖ LSB size
- ❖ Differential nonlinearity (DNL)
- ❖ No missing codes
- ❖ Integral nonlinearity (INL)
- ❖ Accuracy

Additional specifications relevant to static tests are:

- ❖ Conversion time
- ❖ Aperture time
- ❖ Aperture jitter
- ❖ Transition noise

Full scale range (FSR)

The maximum extremes of input signal for an ADC. This parameter varies widely depending on the ADC and can be specified as current or voltage which is positive, negative or both. Common values are $\pm 2\text{mA}$, -2mA , $\pm 10\text{V}$, $\pm 5\text{V}$, 5V . Devices whose input range does not cross through 0 are called *unipolar* while those with \pm input polarities are *bipolar*.

Offset error

Offset error is the difference between the ideal zero point value and the calculated zero point value. This can be described as the actual ADC input value when the first digital output code transition occurs minus $\frac{1}{2}$ device LSB minus the difference between the ideal zero point value. Subtracting $\frac{1}{2}$ LSB puts the end point at "code center" of the analog zero scale input. As an equation,

$$\left(V_{\text{input at zero scale transition}} - \frac{1}{2} \text{LSB} \right) - \text{Ideal zero point value} = \text{Offset error} \quad (4.1)$$

NOTES:

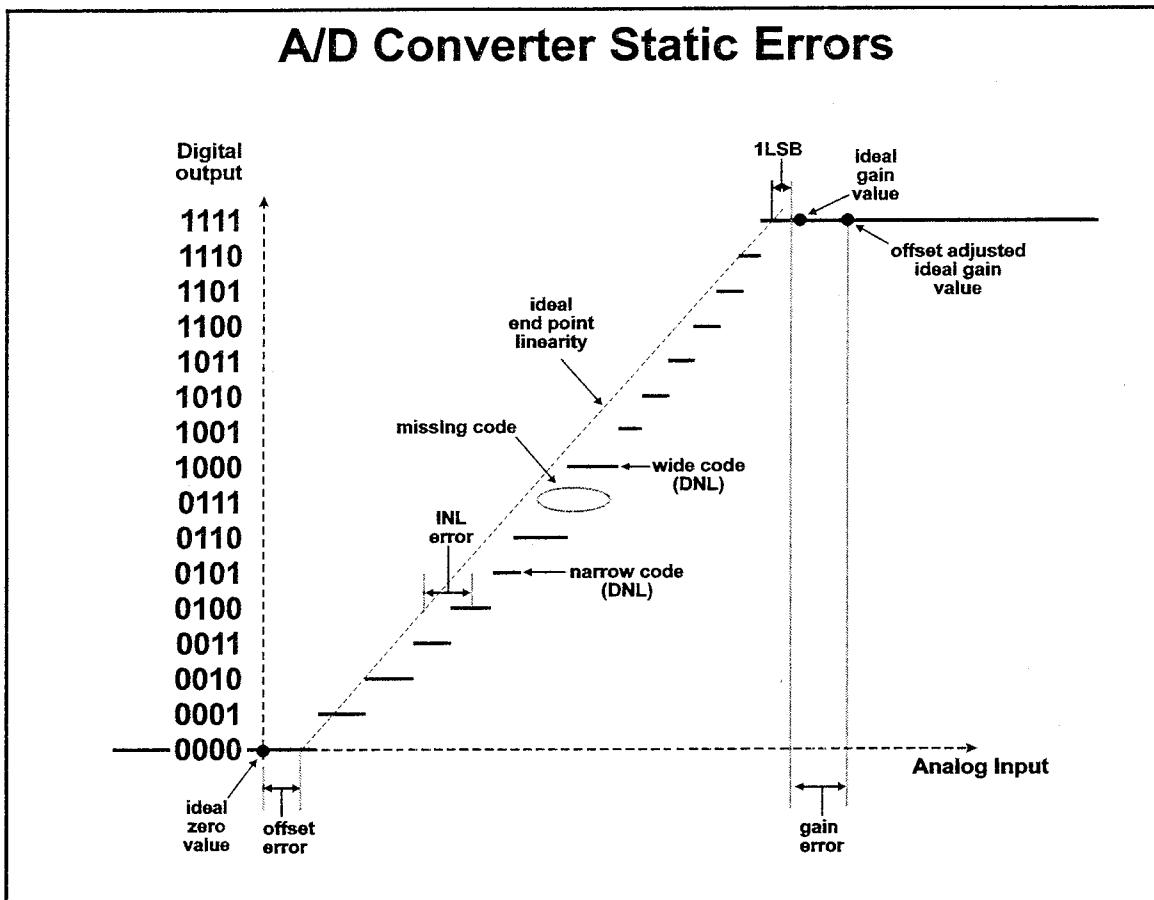


Figure 4.1

Gain error

Gain error is defined as the error in the full scale output of an ADC as calculated from $1\frac{1}{2}$ device LSB greater than the full scale output transition, after compensating for offset error.

LSB size

A linear ADC transfer characteristic is shown as a series of steps along a straight line. A single ADC output results from a 1 LSB wide range of analog input voltage (for an ideal device). The input can be anywhere

NOTES:

within that range and the output will not change. This is different from a DAC, in which a single digital input code always gave a single analog output value. For an ADC, an infinite number of analog input values within an LSB range will give the same digital output. Thus it is impossible to use the digital output from a given input value to get an exact measurement of ADC performance.

The concept above is so important to understand that it bears repeating. *There is an infinite range of analog input values between each code transition point that will produce the same ADC output code.* The logical conclusion which follows is that ***an ADC cannot be tested with only one input value for each expected output code.*** How, then, do we measure ADC linearity?

Figure 4.1 shows a linear transfer characteristic for the 16 output values of a 4 bit ADC. The output increments by one binary count for each increase in the input by one LSB. An LSB for an ADC is defined the same way a DAC LSB is defined, where FSR is the full scale *input* range of the ADC. An ideal LSB is calculated from the specified FSR whereas when testing, an LSB is normalized to the actual length of the transfer curve. In other words it is adjusted for each specific device's offset and gain errors.

Note that there is (ideally, i.e. with a noise free device) a single analog input value at which the ADC output changes from one digital value to the next. This *transition point* gives an exact correlation between analog input and digital output and is used to determine ADC linearity. If we know 2 transition points, we can then calculate the analog input range between which a digital output code will occur. Given 2^N output codes, there are $2^N - 1$ transitions; and this is where the confusion occurs. To calculate the device *LSB size*, measure the lowest and highest transition points. There are $2^N - 1$ transitions with $2^N - 2$ codes between them, meaning there are $2^N - 2$ LSB widths between the 2 measurements. Device average LSB size is thus calculated as:

$$LSB = \frac{Vin_{FST} - Vin_{ZST}}{2^N - 2} \quad (4.2)$$

where Vin_{FST} is the full scale transition point, Vin_{ZST} is the zero scale transition point and N is the number of bits. Adding to the confusion is the usual situation that a 2^N bit converter has outputs which go from 0 to $(2^N - 1)$, e.g. a 12 bit converter has 4096 outputs which range from 0 to 4095.

Differential nonlinearity (DNL)

The ADC transfer characteristic found most often is *linear*, in which the digital output signal has a constant multiplicative relation to the input code. A linear transfer characteristic can be related to the familiar equation for a line, $Y = mX + b$. Other transfer characteristics are, for example, logarithmic output such as found in codecs and "rotational" output of degrees. as found in rotary encoders.

Differential nonlinearity for an ADC is a measure of the "small-signal" linearity error, and is defined as the difference in the analog input range between 2 transitions and a device LSB.

NOTES:



No missing codes

A phrase to state that, with increasing analog input signal, ADC digital output codes increase one digital count for each input LSB change and all possible output codes are produced. No missing codes is guaranteed if DNL is less than ± 1 LSB. It can be specified as a number of bits equal to or less than the resolution of the device, e.g. a 14 bit ADC may be specified as "no missing codes to 12 bits." This indicates that some codes may not appear if the lowest 2 outputs are tested, but that all codes appear for the top 12 outputs. It at first appears that the lower 2 bits are not useful, but statistically they do improve ADC performance.

Integral nonlinearity (INL)

If differential nonlinearity is the small-signal linearity error, integral nonlinearity is the large-signal nonlinearity error. In a sense, integral nonlinearity is the cumulative effect, at any given input, of all differential nonlinearities. It is calculated by first calculating a straight line between the ADC endpoints.

Recall that, for a DAC, any point which deviates from a line between the first and last points is considered a linearity error. Drawing that same line for an ADC is more complicated. ADCs are designed such that the ideal analog input value for a given output code falls in the middle of the analog input range which produces that output code. The line is drawn between $\frac{1}{2}$ LSB below the lowest transition point and $\frac{1}{2}$ LSB above the highest transition point. The LSB used must be the one calculated for the specific device as discussed above. Integral nonlinearity is measured by testing how far a given code center input deviates from this ideal (straight) line drawn between the ADC transfer function end points.

Accuracy

How well an ADC matches a perfect device. This includes all the errors noted above and may be given in percent of reading similar to the way voltmeters are specified. This parameter is not tested explicitly but is implied by the static error parameters.

Other Important Parameters

Conversion time

The time required to convert a single point of an input signal to its digital value. Varies widely among different ADC architectures, generally in msec for integrating converters, (ones or tens of) μ sec for successive approximation and delta-sigma converters, and (tens or hundreds of) nsec for flash converters.

Aperture time

The time required for an ADC to "capture" a point on an analog signal. If an ADC has a track/hold on its input, this is the time required for the track/hold to switch from track to hold mode. If the ADC has no track/hold, this is the same specification as conversion time.

NOTES:

Aperture jitter

The time variation in aperture time between successive ADC conversions. For example, with a sine wave input signal and a convert signal synchronized to the zero crossing of the sine wave, the variation in output over some number of samples reflects the aperture jitter (plus noise).

Transition Noise

Real world noise, including fundamental noise sources such as Johnson noise and shot noise, affects ADC conversion accuracy. Thus on successive conversions of the same constant input signal, the digital output may vary between 2 adjacent output code values. This variation is known as *transition noise*, because it is visible as a fuzzy vertical trace between 2 adjacent output codes on an ADC transfer characteristic graph.

NOTES:

Test System Configuration for ADC Static Parameter Tests

Like a DAC, Figure 4.2 shows that the WD, WG and DSP components of a mixed signal test system are not required for static linearity for an ADC. Instead you see blocks for finding an ADC output code transition; the circuitry inside those blocks will be discussed.

The digital subsection is required for clock, start convert, output capture and other digital functions. Depending on the ATE system, some of the ADC interface circuitry may need to be custom built.

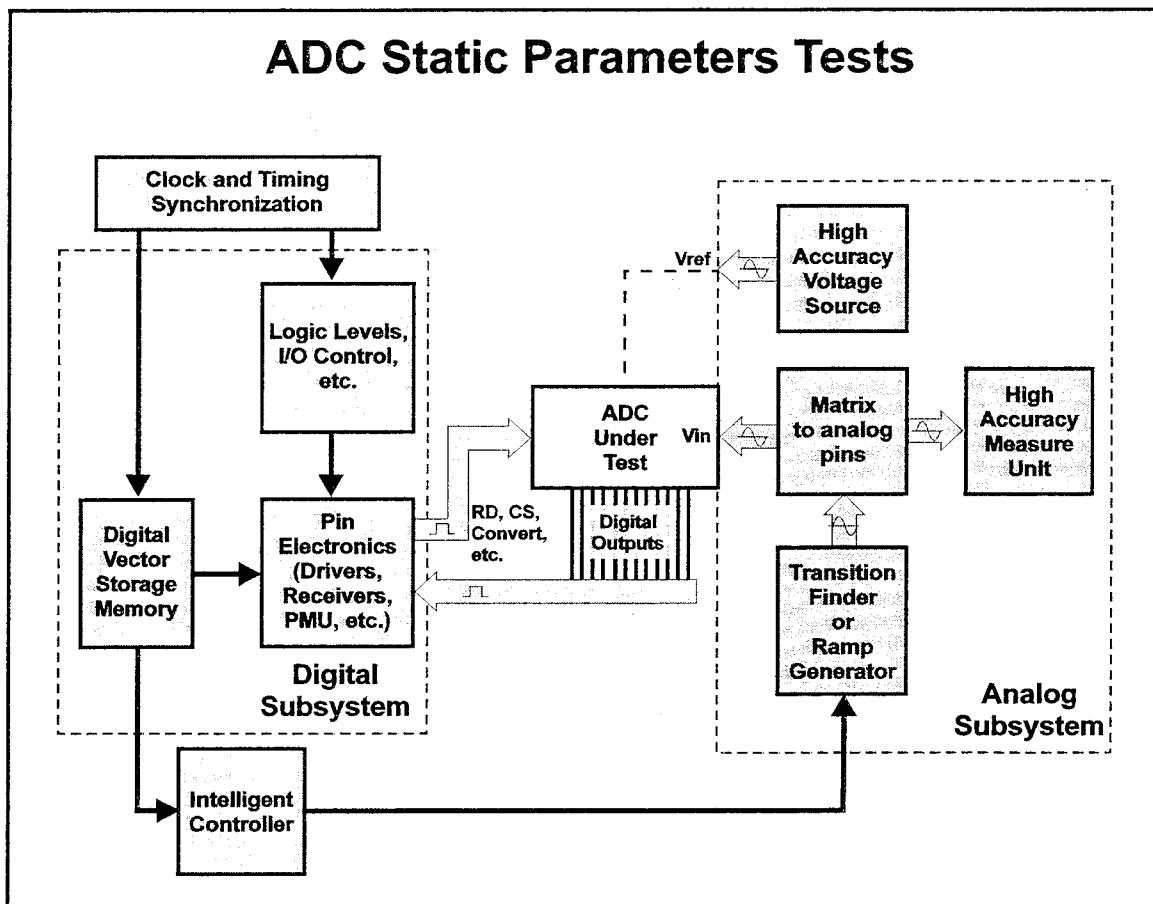


Figure 4.2

NOTES:

Example device specification

The following table is an example of an ADC data sheet specification; note the static error parameters: offset and gain error, differential and integral nonlinearity and input range. Also of interest for test purposes are the resolution and output format.

Parameter	Conditions	Min	Typ	Max	Units
OUTPUT					
Resolution		14			Bits
Output format	straight binary				
STATIC					
Offset error	Output code = 0x0			0.8	% FSR
Gain error	Output code = 0x3FFF			2.0	% FSR
Differential nonlinearity	no missing codes to 14 bits			± 1	LSB
Integral nonlinearity				$\pm 1/2$	LSB
Input range		0 to +4V			V
DYNAMIC					
SNR	$f_{in} = 1000\text{Hz}$ sinusoid	76	80		dB
THD	$f_{in} = 1000\text{Hz}$ sinusoid		-78	-70	dB
IM	$f_{in} = 1000\text{Hz} + 3100\text{Hz}$ tone			-72	dB
AC					
$t_{acquisition}$	After Busy signal goes inactive		200	500	nsec
$t_{aperture}$	After Start Convert signal goes active		10	20	nsec
Conversion time				25	μsec

Table 4.1 ADC example specification

Parameter Measurement Requirements

As noted above, the static parameters are specified relative to the device under test. In other words, they are not measured directly as a voltage or current, but require multiple input stimuli and output measurements. These measurements are used to calculate the parameters that are compared to the specification limits.

NOTES:

Offset and gain error calculations require zero and full scale measurements, which are also needed for DNL and INL measurements. Thus offset and gain are discussed first. (The *Resolution*, *Input format* and *Output range* specifications seen in Table 4.1 give information required testing and are not true test parameter limits.)

Parameter to test	Measurement(s) required	Items needed for measurement(s)
Offset error	zero scale value	<ul style="list-style-type: none"> ◊ input signal resulting in zero scale output code transition ◊ DUT LSB size
Gain error	<ul style="list-style-type: none"> 1. zero scale value 2. full scale value 	<ul style="list-style-type: none"> ◊ input signal resulting in zero scale output code transition ◊ input signal resulting in full scale output code ◊ DUT LSB size
DNL	<ul style="list-style-type: none"> 1. zero scale value 2. full scale value 3. input values for pairs of adjacent output code transitions ("code widths") 	<ul style="list-style-type: none"> ◊ ADC resolution in bits ◊ DUT LSB size ◊ output codes to test
INL	<ul style="list-style-type: none"> 1. zero scale value 2. full scale value 3. input values of center of code for selected output code pairs 	<ul style="list-style-type: none"> ◊ ADC resolution in bits ◊ equation for line between zero scale and full scale ◊ output codes to test

Table 4.2 Measurement requirements for static parameter testing

NOTES:



The Digital Side of ADC Testing

For digital inputs, an ADC must have at the very least a *Start Convert* (SC) signal. You can surmise that when SC goes active, the ADC begins its conversion process. There may be an external clock input if the ADC has no internal clock (depending on converter architecture).

When the conversion process begins, an output signal may occur which we will call *Busy* and indicates that either the output data is invalid or that another conversion cannot be started, or both. The complement of *Busy* could be used as a *Data Valid* signal. After some period of time (the *Conversion Time* as specified in Table 4.1 on page 98), the conversion is complete. The *Busy* signal goes inactive, the *Data Valid* signal goes active and valid data is available at the output pin(s).

Modern converters usually are designed to be compatible with a microprocessor, so the outputs may be in a high impedance state, awaiting activation by a *Read* signal. The data outputs may be configured as *parallel* data (all bits available at once), *serial* data (all data is retrieved from one output, a bit at a time), *byte* data (8 bits available at a time) or *nybble* data (4 bits available at a time). The data sheet for the converter will illustrate how to retrieve the data into the digital pin electronics of the test system.

As discussed on page 104, multiple conversion cycles are required to find a single ADC transition point, thus the timing period shown in Figure 4.3 occurs many times during a complete ADC test, whether taking sine wave samples for dynamic measurements or ramp samples for linearity measurements. The number of conversion cycles is compounded if multiple transition measurements are averaged. This is why it is so important to find a transition as quickly as possible.

NOTES:

1. The term "conversion" is used here to mean the process of digitizing an analog signal. In other contexts, conversion may refer to the process of changing data from one form to another, such as from analog to digital or from one digital format to another.
2. The term "sample" is used here to mean a discrete measurement of an analog signal at a specific point in time. In other contexts, sample may refer to a representative portion of a larger population or a single unit of a product.
3. The term "ramp" is used here to describe a signal that increases or decreases linearly over time. In other contexts, ramp may refer to a signal that increases or decreases exponentially or follows a more complex waveform.
4. The term "linearity" is used here to describe the quality of a signal's waveform. A linear signal has a constant slope, while a non-linear signal has a varying slope. Linearity is often measured by comparing the measured signal to a reference signal.
5. The term "averaging" is used here to describe the process of calculating the mean value of a signal over a period of time. Averaging is often used to reduce noise and improve the accuracy of measurements.
6. The term "transition" is used here to describe a change in the state of a signal. A transition can be a change from one level to another, or it can be a change from one state to another. Transitions are often used to trigger events or to synchronize signals.
7. The term "clock" is used here to describe a signal that provides a timing reference for the conversion process. A clock signal is typically a periodic square wave that is used to synchronize the sampling and conversion processes.
8. The term "data sheet" is used here to refer to a technical document that provides detailed information about a specific electronic component, such as an ADC. The data sheet typically includes information about the component's specifications, operating conditions, and application examples.
9. The term "digital pin" is used here to refer to a physical pin on an electronic component that is used to interface with a digital signal. Digital pins are typically used to connect components to a microprocessor or other digital logic.
10. The term "electronics" is used here to refer to the field of engineering that deals with the design, development, and manufacture of electronic components and systems. Electronics is a broad field that includes topics such as circuit design, signal processing, and system integration.

ADC Digital Interface

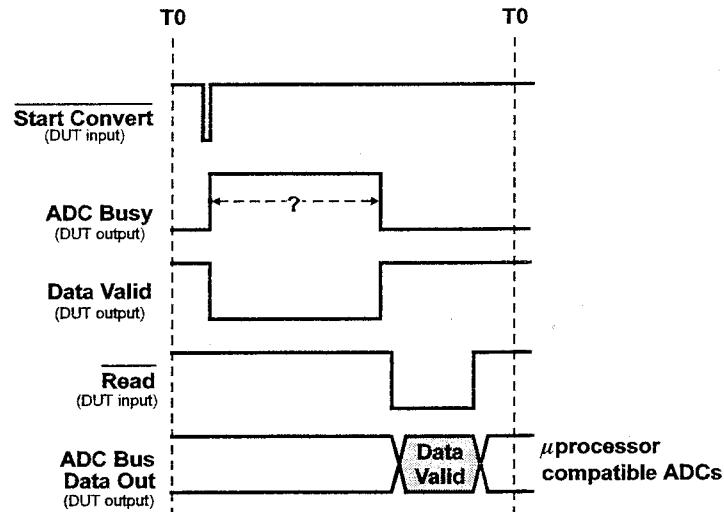


Figure 4.3

Question 4.1: Looking at Table 4.1 on page 98 and Figure 4.3, what is the specification that corresponds to the time period indicated with a mark?

NOTES:



Unique ADC Testing Problems

Transitions, zero scale, full scale and LSB size

In general, testing devices requires creating the proper stimulus to the device then measuring the output response due to that stimulus. The first issue to understand about ADC testing is that testing one that way *does not work*. Testing an ADC requires monitoring the digital outputs while continuously converting, then when the output equals the point we are testing, measure the analog input signal. Conceptually it is sort of backwards from what test engineers are accustomed to doing.

Next, we cannot uniquely characterize an ADC by measuring the input value when the output is static. Measurement of the ADC *transition points* is the only way to get unique values for the ADC transfer curve. The phrase "transition points" indicates that the converter output is in transition, that is, changing from one code to the next adjacent code. Thus one of the problems in testing an ADC is setting its input signal so that the output is changing between 2 codes, and knowing or measuring the exact value of that input signal. Various ways to find a transition point will be discussed.

Another ADC problem is the often confusing situation regarding ADC endpoints, FSR and LSB size. Since zero and full scale *transition points* (not the true end points) are the values which can be distinctly measured, there are $2^n - 2$ LSBs between the zero and full scale transitions. For example, in a 3 bit converter as shown in Figure 4.4, you can see 8 ($=2^3$) horizontal steps in the transfer plot, with 7 transitions separating them. There can only be 6 steps (output codes) between 7 transitions and we have stated that only transition points can be uniquely determined.

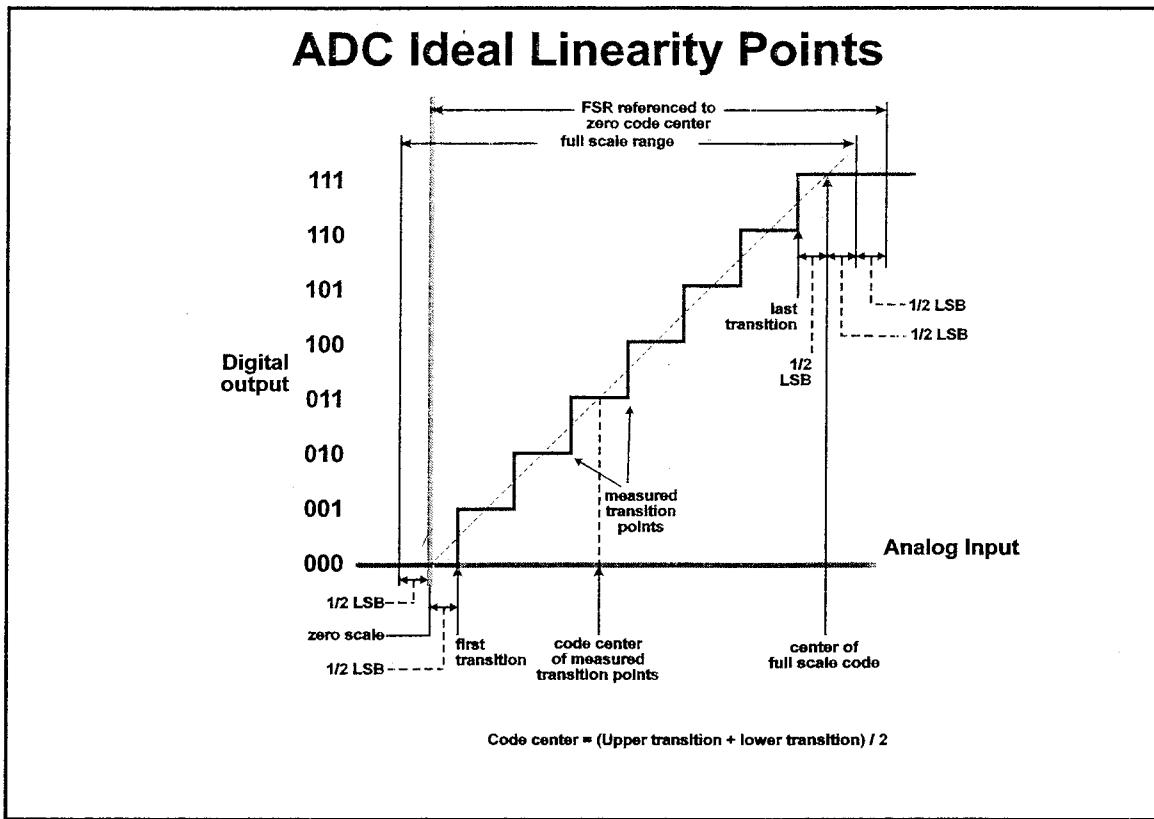
The first step is to calculate the LSB weight for the DUT. An average device LSB for an ADC is defined as the input range between the zero and full scale *transitions* divided by the number of LSBs between the 2 transitions:

$$LSB_{Device} = \frac{Vin_{fst} - Vin_{zst}}{2^{bits} - 2} \quad (4.3)$$

where *fst* = *full scale transition* and *zst* = *zero scale transition*.

There are 2^{bits} digital output levels, the first of which is zero (for our straight binary ADC). To optimize linearity, ADC manufacturers specify linearity referenced to the point centered between each transition, called the *center of code*. Ideally, the input signal must move $\frac{1}{2}$ LSB in either direction from center of code to cause the output to change state. To make this true for the zero scale center of code, that point is defined to be $\frac{1}{2}$ LSB weight below the output's first transition.

NOTES:

**Figure 4.4**

Referring to Figure 4.4, notice that this causes the zero scale transition to be $\frac{1}{2}$ LSB above the starting point for the linearity transfer curve. This gives us an equation for calculating zero scale input based on zero scale transition measurement and calculated LSB size:

$$Vin_{zs} = Vin_{zst} - 0.5LSB_{Device} \quad (4.4)$$

Full scale range (FSR) is defined as the input range between the zero and full scale transitions plus 2 LSBs to include an LSB of input range each for the zero code and the full scale code:

$$FSR_{device} = Vin_{fst} - Vin_{zst} + 2LSB_{Device} \quad (4.5)$$

NOTES:

The input value that represents the actual full scale output center of code point is found by subtracting $\frac{1}{2}$ LSB from the full scale transition to compensate for the $\frac{1}{2}$ LSB shift of the line below the zero scale transition, then adding back the 2 LSBs of input range for the zero and full scale output codes:

$$Vin_{fs(actual)} = Vin_{fst} - 0.5LSB_{Device} + 2LSB_{Device} \quad (4.6)$$

resulting in the actual full scale input value (*not* the FSR) that is 1.5LSBs above the measured full scale transition:

$$Vin_{fs(actual)} = Vin_{fst} + 1.5LSB_{Device} \quad (4.7)$$

Gain error is the difference in the FSR_{device} and the *Output Range* given in the Table 4.1 on page 98. Thus with 2 measurements of Vin_{fst} and Vin_{zst} and knowing the ADC resolution, we can calculate the LSB size and the end points of the INL line (Vin_{fs} and Vin_{zs}), which also gives us the values necessary for calculating offset and gain error.

Question 4.2: What state does the ADC output code take when the input signal goes above full scale?

ADC transition points

With a firm grasp on how to calculate a "DUT LSB" for an ADC, the next question is how to find the input value at which an ADC's output toggles between 2 adjacent codes on successive conversions. The words "successive conversions" are critical to understand—during an ADC test, the device is being clocked and is continuously converting its analog input signal into digital output codes. Multiple ADC conversions are required to make each transition measurement. Test time depends on how quickly you can find and accurately measure a transition point and how many transitions must be measured.

Using a D/A to step the ADC input

Perhaps the most obvious way to find a transition is to put a signal into the ADC which is below the transition of interest, then, monitoring its output while the ADC continuously converts, gradually increase the signal until the ADC output code changes. As long as the input can be changed in small enough steps and the input is known, this is a valid way to find a transition.

NOTES:

A/D Transition Search with Stepped D/A

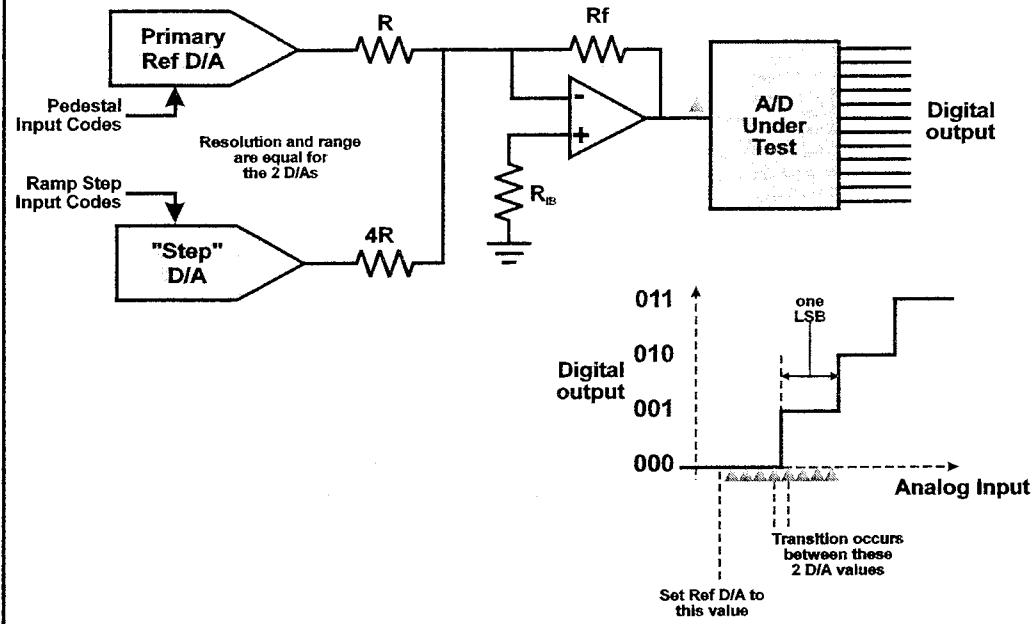


Figure 4.5

Question 4.3: In Figure 4.5, the *Ref D/A* and the *Step D/A* are identical voltage out DACs and have been calibrated to perfect accuracy. They have FSR = 10V and resolution = 12 bits; assume $R_f = R$. **a)** How many LSB steps will the *Step D/A* make for each *Ref D/A* step? **b)** What determines this? **c)** What is the closest ADC Vin measurement accuracy possible by knowing the *Step D/A* input code?

NOTES:

A/D Transition Noise and Hysteresis

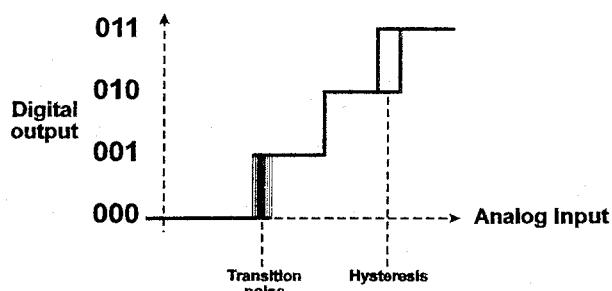


Figure 4.6

Finding the transition point this way does not, however, completely characterize the ADC at that code. Some ADCs exhibit *hysteresis*, in which the transition occurs at a different input value when the input is increasing versus when the input is decreasing. For example, the change from code 000 to code 001 occurs at $V_{in} = 0.1V$ but the transition from code 001 to 000 occurs at $V_{in} = 0.05V$. Converters may also have *noisy* transitions, in which the input value that changes the output code may vary from conversion to conversion by as much as $\frac{1}{4}$ to 1 or more LSBs. Both of these effects may be compensated with averaging (discussed later).

NOTES:

Calculating Offset Error

We begin this section by assuming that we have successfully created a DUT hardware and software configuration on the test system which will find a transition edge. It makes no difference whether we use an input ramp, a dither loop or some other technique; we ultimately end up with a voltage (or current) value which, when used to drive the ADC input, toggles the output between all zeros and all zeros plus one (e.g. $00000000 \leftrightarrow 00000001$ for an 8 bit ADC). This is again using a *unipolar straight binary* output converter to keep things simple.

Once we have this number, what do we do with it? As it turns out, with this number alone we can do nothing. Recall from the discussion on ADC end points on page 102 that the zero scale value is defined as $\frac{1}{2}$ DUT LSB less than the zero scale transition point. Before we can calculate the zero scale value, the LSB size must be known, requiring a measurement of the full scale transition point. OK, assume we have also located and measured the input value which toggles the outputs at the full scale transition point. With both of these values, calculate the LSB size as indicated in Equation (4.3) on page 102.

Question 4.4: The 14 bit ADC as specified on page 98 has a zero scale transition measurement of 0.024092V and full scale transition measurement of 4.107253V. **a)** What is the device LSB size? **b)** What is the actual zero scale value? **c)** What is the offset error? **d)** Does this device pass the offset error limit? [If you need a calculator, use the one in the DSP Lab software.]

NOTES:

Calculating Gain Error

Gain error is calculated from our zero and full scale transition point measurements and subtracting that from the ideal FSR minus 2 device LSBs. In other words, the ideal full scale range is adjusted by 2 device LSBs so it can be compared to the measured range between the device's first and last transitions.

After calculating offset error, all required information to calculate gain error is available. Using the previous example values, we know these things already:

1. Full scale transition point = 4.107253V
2. LSB size = 249 μ V
3. Zero scale transition point = 0.024092V

The full scale range is defined by Equation (4.5) on page 103, repeated here:

$$FSR = Vin_{fst} - Vin_{zst} + 2LSB_{Device} \quad (4.8)$$

This evaluates to $4.107253 - 0.024092 + 2*249 \times 10^{-6} = 4.083659$

Subtract the ideal FSR value (= 4V) from this to get an error of 0.083659V

The specification states that gain error limits should be 2%FSR, resulting in limits of ± 0.08 V. This device is out of specification for gain error by over 3mV.

NOTES:



DNL and INL Testing

There are several methods of taking sample set data for ADC linearity measurement and, in one way or another, they are all based on statistics. Differential and integral nonlinearity can be measured on an ADC by a technique known as "histogram" testing or by using one of the transition finding techniques such as a dither loop, both of which require multiple measurements for a single ADC output code. If a high accuracy digital voltmeter (DVM) is used to measure the input value to produce a given ADC output code, the DVM takes several (or several tens of) milliseconds to make its measurement; multiple ADC conversions must occur during this time. The DVM is actually measuring an average of the ADC input. With all of these measurement techniques requiring multiple ADC conversions, static INL/DNL is always a statistical measurement. The first method we consider is the histogram, then we discuss a closed loop method.

Histogram Test for DNL and INL

The ramp method uses a linearly increasing or decreasing signal as the input signal to the ADC under test. It is similar in nature to the stepped input method pictured in Figure 4.5 on page 105. The idea is to change the ADC input in a series of known small steps, *small* being defined as "significantly smaller than a DUT LSB." As the DUT input changes, the output changes state from the current output code to the next. If we know the DUT LSB size and the step size of the input change, we should be able to predict how many input step changes are required to change the DUT output state.

Figure 4.7 illustrates an ADC transfer graph with a DAC input step superimposed on it. This is slightly confusing because the DAC axes are different from the ADC axes, but hopefully the concept is clear. For each ADC output step there are 16 different input values.

NOTES:

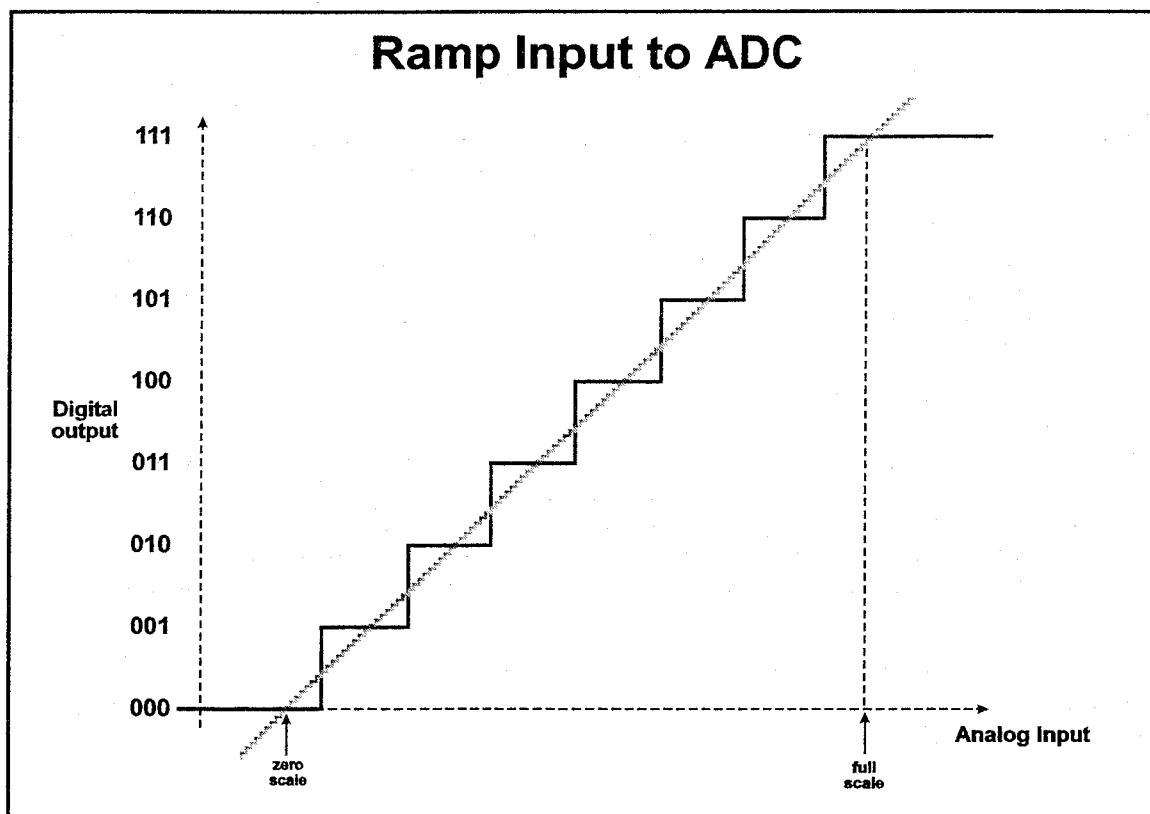
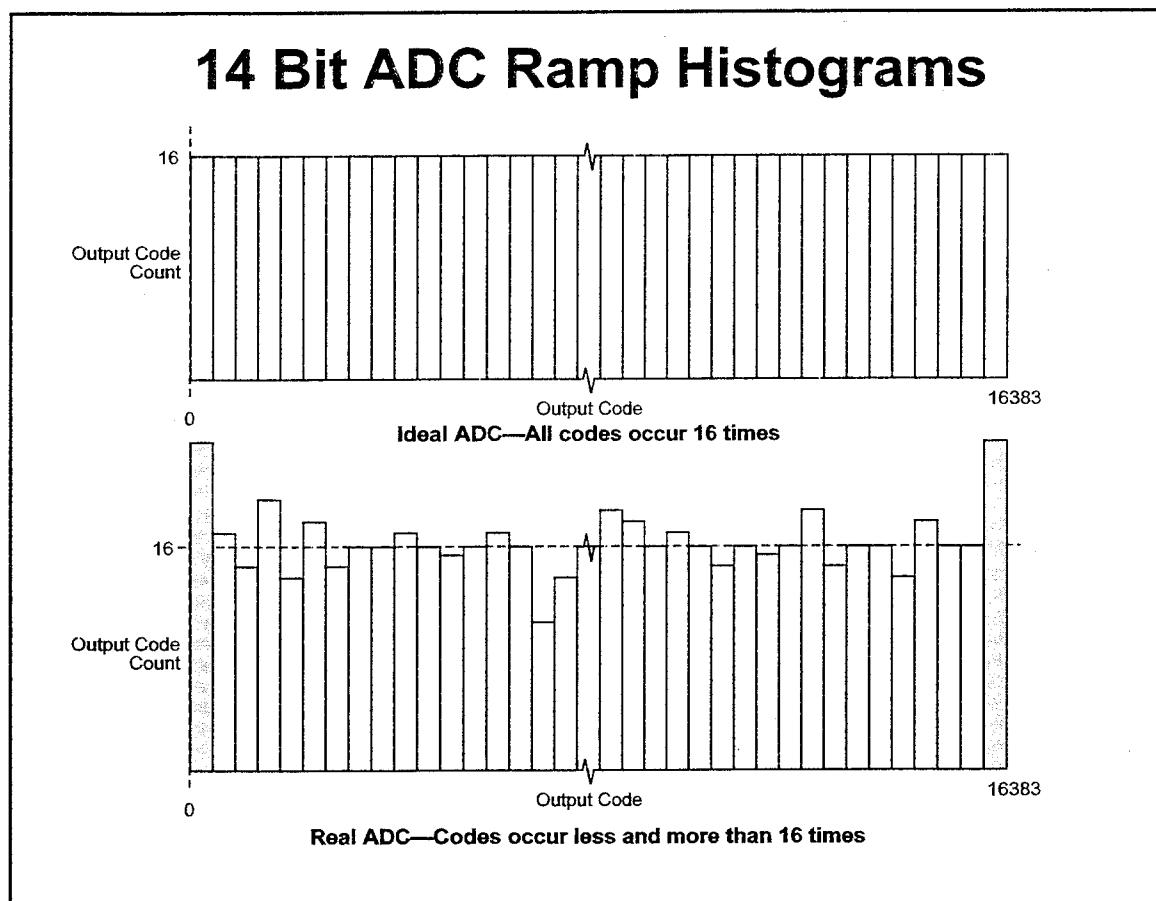


Figure 4.7

Question 4.5: Given a 14 bit ADC, how many additional bits does the DAC need to drive the ADC input with 16 input steps per output code, if both have equal full scale ranges?

NOTES:

**Figure 4.8**

An ideal DUT would have 16 output codes of the same value, then 16 of the next value, etc. A statistical distribution of this would look like a rectangle of height 16 and width 2^{bits} . A real device would have some “wide codes” with a count of more than 16 and some “narrow codes” with a count of less than 16. The real device should, however, have a sum total of all occurrences of all codes which equals 2^{bits} times 16. The example shown in Figure 4.8 is an example of ideal and real results of a 14 bit ADC with its input coming from an 18 bit ramp DAC. The input DAC has 4 more bits of resolution, giving $2^4 = 16$ input steps per ADC transition.

NOTES:

Notice in Figure 4.7 and Figure 4.8 that the input goes below zero scale and above full scale. This is necessary to ensure that the full input range is covered by the ramp and requires that the code counts at 0 (zero scale) and 16383 (full scale) be ignored in the calculations.

The ramp input method can also be implemented using a *pedestal DAC* and a *ramp DAC*.¹ The pedestal DAC sets a point along the transfer curve and the ramp DAC moves the input in small increments. The pedestal DAC sets the segment of the input signal and the ramp DAC moves the input above and/or below that segment. This allows much finer resolution of the input step but requires that an overlap occur at the end of one segment and the beginning of the next. This causes "cumulative INL errors due to segmentation"² making the implementation and the math more complicated. This technique is exactly the same as pictured in Figure 4.5 on page 105, where the pedestal DAC is labelled "Primary Ref D/A" and the ramp DAC is labelled "Step D/A."

One interesting aspect of histogram testing is that it clearly shows when a code is missing. A missing code is a possible ADC output state which never occurs. This immediately disqualifies a DUT from being considered an *n bit device*. For example, if a 14 bit ADC has a missing code, it can no longer be considered 14 bits linear and must be sold as either an 13 bit device or tossed in the garbage. A code position in the histogram with a vertical bar height of zero means a missing code at that output location. Ramp histogram testing does not find *sparkle codes* (also called *spurious codes*). This requires a more complex histogram method using a sine wave, which is covered in chapter 8.

Note that using the histogram technique gives no absolute information about ADC input values versus output code. To measure zero and full scale values (and their corresponding offset and gain errors), some other measurement technique (e.g. a precision meter) must be used.

Calculating DNL from the histogram

Differential nonlinearity is the difference in the expected code count and the actual code count for each ADC output code (ignoring the zero and full scale codes). In the 16 count per code example given above, if a particular output code has a count of 21, it has $(21 - 16) / 16 = 5/16$ LSB DNL.

Calculating INL from the histogram DNL data

Integral nonlinearity is the "area under the DNL curve." The DNL histogram is built from rectangles, so the area under the curve is the summation of the DNL values for each code. For a 14 bit ADC, the calculation is given in the following pseudo-code:³

```
float INL[16384]; /* Holds INL error value calculations */
INL[0] = 0; INL[16383] = 0; /* End point linearity defines this */
for Index = 2 to 16382 do
    INL[Index] = INL[Index - 1] + (DNL[Index] + DNL[Index - 1]) / 2;
```

NOTES:

Using an intelligent feedback loop to test DNL and INL

To accurately find the input value that causes an ADC output to transition between 2 adjacent states, a feedback loop is used to move the input signal to an exact transition point. The difference in the ADC output and a desired output state can be calculated in an intelligent controller as seen in Figure 4.9. The difference in ADC output and desired code is used to set the ADC input signal via a DAC. The output error is decreased on each successive conversion until eventually the ADC output toggles between the desired output and one adjacent output state. With a calibrated *Dither DAC* and *Level Setting DAC* the input signal is known from the digital values used to set the DACs, meaning that no WD, meter nor PMU is required.

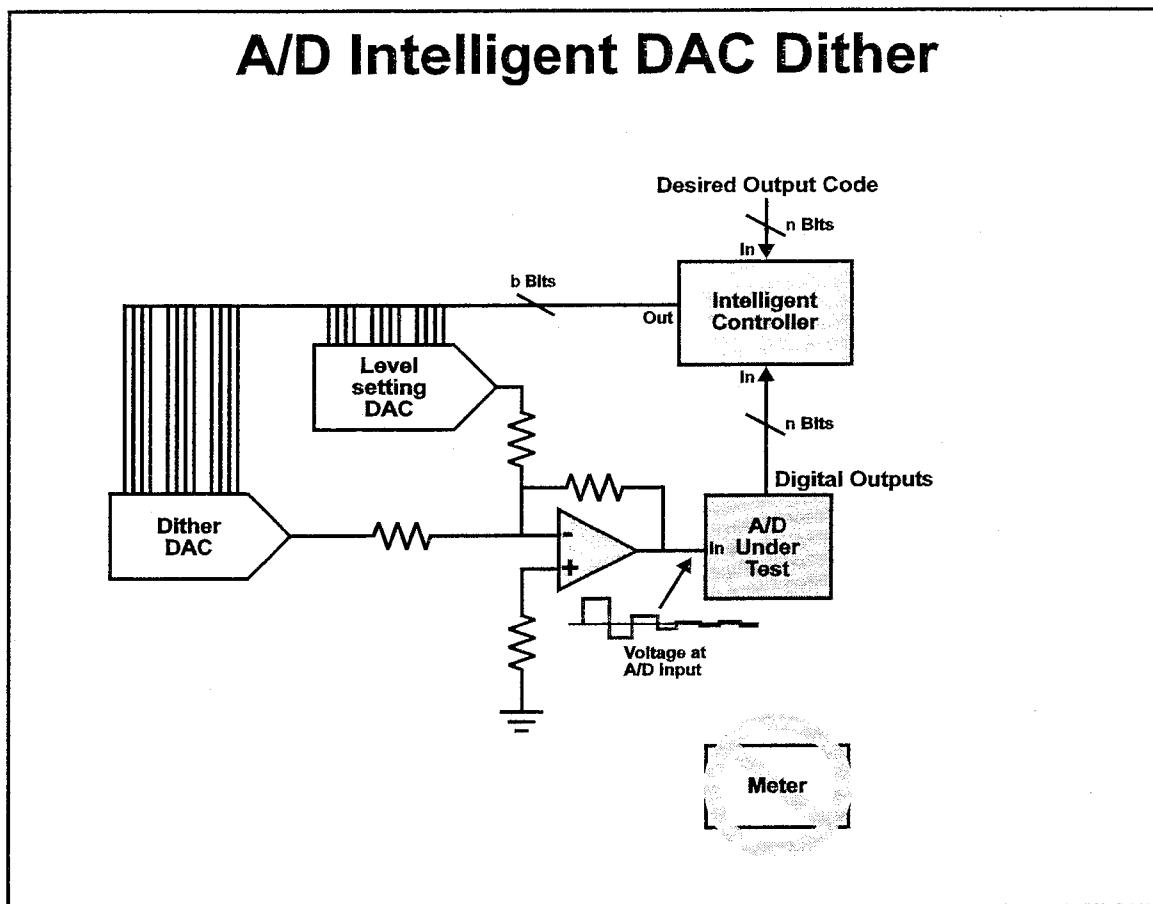


Figure 4.9

NOTES:



This type of feedback loop was initially used in a bench setup with a voltmeter, integrator circuit and digital comparison and was subsequently adapted to ATE. It has been used successfully by ATE companies and their customers to make fast, accurate and repeatable ADC linearity measurements.² A binary search algorithm will provide a rapid convergence to the optimum input value for a given code transition. Given a small enough incremental input change from the dither DAC, the input signal can be made to dither over a range equal to the transition noise for a specific output code pair.

Which codes to test?

The biggest hurdle in measuring DNL and INL is collecting the data. The collected data is a set of transition points which may include all ADC output codes or only a relevant subset of all output codes. Which codes to be tested depends, as with DAC testing (as discussed in *Important input codes* on page 75), on the ADC architecture. A table of architectures and whether to test all or some codes is below:

Architecture	Codes to test
Flash converter	All codes
Successive Approximation Register (SAR) types, using a resistor or capacitor ladder DAC	Depends on internal DAC—at least major transitions and any decoded upper bits
Delta sigma	DNL is guaranteed by design; INL is limited by physical elements in circuit.
Subranging or “two pass flash”	Depends on ADC design—consult with manufacturer or design engineer
Algorithmic or “cyclic”	Generally the same as SAR

If in doubt as to the codes to test, you can always fall back on the “test all codes” philosophy, although you should not make this decision casually. It can cost significant money in DUT test time to test 65536 codes of a 16 bit ADC when you could instead be testing only 16 or 24 codes. It may well make the difference in device profitability. If you set up a fast transition finding system such as was discussed in *Using an intelligent feedback loop to test DNL and INL* on page 113, it may be reasonable to measure all codes; this is often the case when converter manufacturers test devices of 12 bits and under. Depending on the ADC architecture, it may be necessary to test all codes to ensure that internal decode logic operates correctly.

NOTES:

Finding the “center of code”

When measuring offset and gain error, the zero and full scale transitions were found, the difference between them calculated, then the DUT zero point was calculated as $\frac{1}{2}$ LSB below zero scale transition and the DUT full scale input point calculated as $\frac{1}{2}$ LSB above the full scale transition. Finding points for ADC linearity measurement requires a similar calculation because linearity is measured at *center of code*. That is, the calculated measurement is halfway between 2 adjacent transitions.

With a DAC, recall that differential nonlinearity (DNL) measurement requires a *code width* calculation based on the difference between two adjacent codes. DNL for an ADC is similar in that it requires a code width calculation based on the difference between two adjacent *transitions*. The center of code is halfway between those 2 transitions, meaning the INL data points are different from either of the DNL measured transition points, although the INL point is a simple calculation using the 2 DNL points.

Figure 4.4 illustrates the center of code concept—measure the input value for two adjacent output transitions. Use the average of several transition measurements if necessary. The center of code point is the specific code width divided by 2 and added back to the lower code value. It is also the average value of two adjacent transitions as given by the equation in Figure 4.4.

NOTES:



Static DNL

Referring back to Figure 4.1 on page 93, subtract one transition point from the other to get code width. DNL is the difference between that specific code width and the calculated average DUT LSB width. Mathematically:

$$DNL = (V_{in2} - V_{in1}) - LSB_{device} \quad (4.9)$$

where V_{in2} is the upper transition, V_{in1} is the lower transition. The result is in DUT input units, voltage or current. If necessary, it is converted to %FSR or LSBs for comparison to the specification limit.

NOTES:

1. The DNL calculation is based on the assumption that the device has a uniform quantization error across its full range. This is often not the case, especially for low-resolution converters where the quantization noise is relatively large. In such cases, the DNL values may vary significantly from one code word to another.
2. The DNL calculation does not take into account any non-linearity or distortion present in the converter's transfer function. These factors can contribute to additional errors and noise, which may affect the overall performance of the converter.
3. The DNL calculation assumes that the input voltage is constant during the conversion process. In reality, the input voltage may change over time, which can lead to errors in the DNL measurement. To mitigate this issue, it is common to perform multiple measurements at different input voltages and then average the results.
4. The DNL calculation does not consider the effects of temperature, supply voltage, and other environmental factors on the converter's performance. These factors can also contribute to errors and noise, which may affect the DNL measurement.
5. The DNL calculation is typically used to compare the performance of different converters or to verify that a converter meets its specified performance requirements. It is important to note that the DNL value alone does not provide a complete picture of a converter's performance, and other metrics such as SNR, THD, and SFDR should also be considered.

Static INL

Static INL uses the center of code found earlier. INL can be confusing because the specification only states a single error value, whereas the test itself computes numerous INL values depending on how many codes must be tested. The INL for each tested output code must be compared to the limit and the worst case value stored as the INL for that unit. It is analogous to testing input leakage for many pins and comparing them to the same error specification. As soon as an INL value exceeds the limit, the test can stop and the device is binned as a bad device.

Calculation of INL compares a code center value to the value it should be as determined by the line between the DUT endpoints.

$$INL = \left[\left(\frac{Code}{2^{bits} - 1} \right) x(Vfs - Vzs) + Voffset \right] - Vcode \quad (4.10)$$

where Vfs = center of code full scale input voltage, Vzs = center of code zero scale input voltage, $Code$ = the digital output code being tested and $Vcode$ = the code center input value calculated for this output code. Note that the value within square brackets [] is the ideal distance along the x (input) axis for the code, normalized to the DUT end points. Subtracting the measured distance of $Vcode$ gives the error.

Notice that equation (4.10) does not multiply the LSB value times the output code weight; instead, the ratio of code out to full scale code is used to calculate the DUT-referred ideal point on the transfer line. Using the LSB size requires an LSB to be multiplied by the output code being tested. If there is some slight roundoff or measurement error in the LSB value, that error gets multiplied by a larger and larger DUT code value as it moves toward full scale, resulting in a substantial error in the calculation.

At issue is the order of calculation—first calculate the ratio of output code to full scale code, then multiply this ratio by the FSR as given by ($Vfs - Vzs$). This calculation order multiplies a large value by an increasing fraction rather than multiplying a small value by an increasing large number.

NOTES:

Effective Number of Bits (ENOB)

ENOB is a way to infer the equivalent number of bits of linearity of an ADC from an SNR measurement. It is given by:

$$\text{Bits} = (\text{SNR}_{dB} - 1.76)/6.02 \quad (4.11)$$

Because this topic is more closely related to SNR than to ENOB, it is covered in depth in chapter 8 on page 297.

NOTES:

Key Points of this Chapter

- ❖ ADCs require a way to find a specific output code pair then measure the input value that produces that code. This is the reverse of normal device testing.
- ❖ A range of input values to an ADC will generate the same output code. The only way to find a singular correlation between ADC input and output is to find the input values that cause the output to change from one digital code to another. This is known as "finding the transition."
- ❖ Finding the LSB size of an ADC requires finding the zero and full scale transition points and dividing that input range by $2^{bits} - 2$.
- ❖ ADCs have transition noise.
- ❖ In general, many ADC conversions occur to measure a single input versus output point.
- ❖ Endpoint linearity for INL testing requires a straight line between the "center of code" points $\frac{1}{2}$ LSB below the zero scale transition and $\frac{1}{2}$ LSB above the full scale transition.
- ❖ Dither testing can provide fast transition measurements. This method requires feedback from the DUT output to its input. It may require a fair amount of load board hardware and sophisticated software algorithms, depending on the ATE system.
- ❖ Histogram testing provides a fast and fairly simple way to test DNL and INL. DNL comes directly from the histogram data and INL is inferred from DNL. Measurement accuracy is limited to the number of input steps per DUT code width for a given input ramp.

NOTES:

Answers to chapter questions

Answer 4.1: Conversion time

Answer 4.2: Outputs remain at all 1's.

Answer 4.3: a) 4 b) the current through the 4R resistor from the *Step D/A* is 4 times less than the current through the R resistor from the *Ref D/A*, thus its contribution to the voltage across R_f is 4 times smaller c)
 $\frac{1}{4}\text{LSB} = 10\text{V} / (2^{12} - 1) \times \frac{1}{4} = (2.44\text{mV} / 4) = 610\mu\text{V}$.

Answer 4.4: a) Full scale - zero transition = $4.107253 - 0.024092 = 4.083161\text{V}$

Divide this by $2^{\text{bits}} - 2$ (Equation (4.3) on page 102) to get

$$1 \text{ LSB} = 4.083161\text{V} / (2^{14} - 2) = 4.083161\text{V} / 16382 = 249\mu\text{V}.$$

b) The zero scale measurement is $\frac{1}{2}\text{LSB}$ below the zero scale transition point:

$$0.024092 - 0.000125 = 0.023967\text{V}$$

c) Offset error is the difference in the zero scale value and the ideal offset point:

$$0.023967\text{V} - 0 = 0.023967\text{V}$$

d) Checking against the offset error limit of 0.8%FSR given in Table 4.1 on page 98:

$$\text{Ideal FSR} = 4\text{V}; 0.8\% = 0.008; 0.8\% \text{ FSR} = 4 * 0.008 = 0.032\text{V}$$

The offset error limits are $0 \pm 0.032\text{V}$

Offset error is within specification limits, so this device passes the offset test.

Answer 4.5: 4 times as many bits

NOTES:



References

1. Solomon Max, *Fast Accurate and Complete ADC Testing*, Proceedings of the IEEE International Test Conference, 1989, pp 111-117.
2. Jack Weimer et al, *A Rapid Dither Algorithm Advances A/D Converter Testing*, Proceedings of the IEEE International Test Conference, 1990, pg 501.
3. Matthew Mahoney, *DSP Based Testing of Analog and Mixed Signal Circuits*, IEEE Computer Society Press, 1987, pg 139.

NOTES:

Analog to Digital Converter Static Measurements

NOTES:

The Mathematics of DSP

Goals

- ❖ Review mathematical concepts used in analog signal analysis (logarithms, dB, etc.)
- ❖ Review the basic mathematical concepts used in DSP, including some trigonometry
- ❖ Show how a continuous waveform can be described as a sum of sinusoids
- ❖ Explain the transformation of time to frequency
- ❖ Discuss the basis of complex numbers and their conversion between rectangular and polar formats

Objectives

- ❖ Provide a mathematical basis of understanding necessary for discussion of DSP based testing.



What you will learn

Digital Signal Processing (DSP) requires a lot of mathematical calculations. Our discussion will start with some fundamental principles and build gradually on that foundation. Keep in mind that it is not necessary to understand how all the math is derived. The math should be regarded as a tool to accomplish the needs of the mixed signal test engineer. When this chapter has been completed, you should be comfortable with the concepts behind DSP.

We will discuss

- ◊ Logarithms and log bases
- ◊ What decibels (dB) represent and how they are calculated
- ◊ The relationship between time and frequency
- ◊ How a complex waveform can be created with a Fourier series
- ◊ Complex numbers and converting between rectangular and polar forms

NOTES:

Logarithms and exponents

Logarithms ("logs") and exponents are part of the mathematical grease that makes the discussion of electronics (and other engineering, scientific and natural phenomena) move more easily. A logarithm describes the relationship between a *number*, an *exponent* and a *base* such that a logarithm is the exponent (or power) to which the base must be raised to produce the number¹, e.g. if

$$\text{base}^a = b \text{ then } \log(b) = a \quad (5.1)$$

where the log is in the given base. The base and the exponent can be any real or complex number, rational or irrational. Logs are a type of mathematical transform which have the special property of reducing power operations to multiplication or division and reducing multiplication and division to addition and subtraction. We will examine these operations a little later.

Logarithm bases

The base can be any number you choose. Three bases are in very common use in DSP—

1. the *common* log uses base 10
2. the *natural* log uses base $e = 2.718281828\dots$
3. the *binary* log uses base 2

Common log

The common log of base 10 was originally used to describe the response of the human ear to sound, specified in *decibels*, or dB (deci as in 10, e.g. decimal). The common logarithm is also an easy way to visualize what a log represents. With 10 as the base, we know that $10^1 = 10$ and $10^2 = 100$, so the common logarithm of all numbers between 10 and 100 must be between 1 and 2. Likewise the common log of all numbers between 100 and 1000 must be between 2 and 3, etc.

Another well known common log usage is the Richter Scale for earthquake magnitude. A change of one unit of the Richter Scale is a difference of 10 times greater (or less) magnitude in the power in the earthquake. Logarithms turn large spans of numbers into smaller "orders of magnitude," allowing graphs of large differences in scale to be plotted in a small area. A graph of diode current versus voltage is a good candidate for a log plot because the current changes several orders of magnitude over a few tenths of a volt across the diode.

NOTES:

Logarithm: $y = \ln(x)$

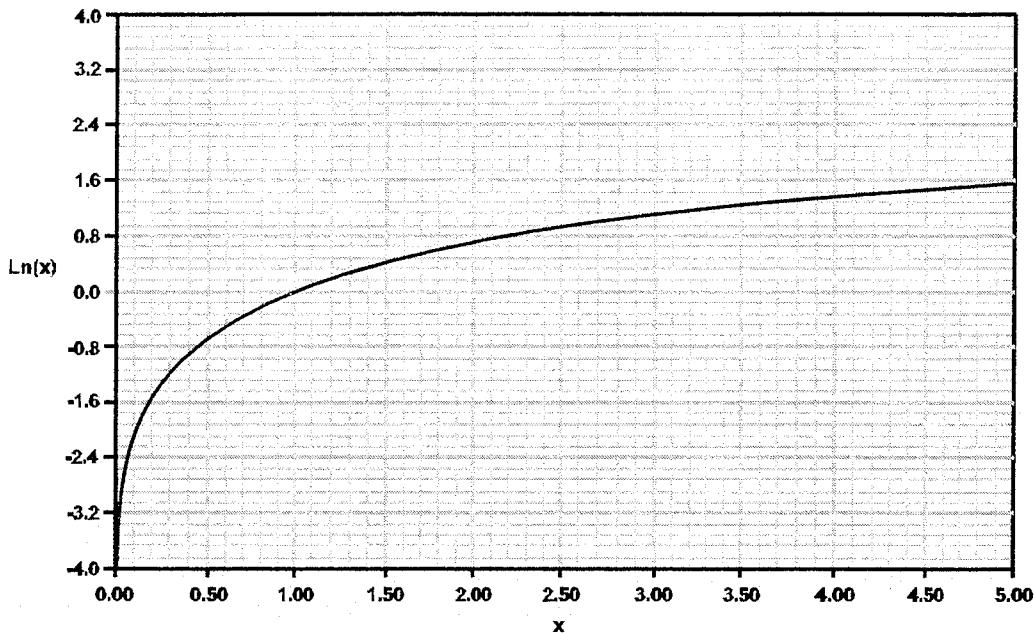


Figure 5.1

Natural log

The base of the natural log is the point of convergence of exponential growth. The base number is an irrational number and does not terminate nor repeat. Thus a special symbol e makes for easier representation of the value so we do not have to write an infinite number of digits all the time. To a few decimal places, $e = 2.718281828\dots$ and even though it looks like the decimal repeats, it does not (the next 3 digits are 459.) You can calculate the value of e with a scientific calculator by entering 1 and pressing the e^x key or the *Inverse Ln* key (because e^x is the inverse function of the natural log of x , abbreviated $\ln(x)$.)

NOTES:

Where does this strange number e come from? It is the value to which the limit of compound interest calculations converge; many *natural* phenomena follow the exponential curve described by e^x —diode current, avalanches, capacitor voltage et al. The *real* magic of e^x is that it is its own derivative and is used in the solution of differential equations. Because e comes up so often in science and engineering, its use as the base of the natural log is, well, natural.

Log base 2

The number 2 is sometimes used as a base for logarithms in DSP because so often the numbers which represent analog samples are binary. Using $\log_2(x)$ makes computations with exponent of 2 numbers easier.

Properties of logarithms

The following properties are true for logarithms of any base.

- ◊ Exponentiation becomes multiplication or division:

$$\log(b^a) = a \log b \quad (5.2)$$

$$\log(\sqrt[a]{b}) = \log(b^{1/a}) = \frac{\log b}{a} \quad (5.3)$$

- ◊ Multiplication becomes addition:

$$\log(ab) = \log a + \log b \quad (5.4)$$

- ◊ Division becomes subtraction:

$$\log\left(\frac{a}{b}\right) = \log a - \log b \quad (5.5)$$

- ◊ Log(1) = 0
- ◊ The logarithms of values from “0+” to 1 are negative
- ◊ Log(0) is undefined; its value approaches $-\infty$ at 0.

NOTES:



Conversion between bases

Calculators generally take logarithms in base e or 10 and it is often necessary to calculate a logarithm in a different base such as 2. This is easily done (using base 10) as follows:

$$\log_b x = \frac{\log_{10} x}{\log_{10} b} \quad (5.6)$$

where b = the different base. For example,

$$\log_2(65536) = \frac{\log_{10}(65536)}{\log_{10}(2)} = \frac{4.816}{0.301} = 16$$

This will also work if base e is substituted for base 10.

Question 5.1: Using a scientific calculator, calculate $\log_2(4096)$.

Question 5.2: Using a scientific calculator, calculate $68 * 777$ using \log_e .

NOTES:



Decibels (dB)

A decibel (usually called dB) is a dimensionless unit of measurement of power ratio. When measuring the power of a signal versus the power of noise contained in that signal, the values are usually several orders of magnitude apart. The dB scale uses the common log as discussed on page 125 to "linearize" this large range of values, making small levels look bigger and large levels look smaller. A definition of dB is

$$dB = 10\log\left[\frac{P_2}{P_1}\right] \quad (5.7)$$

which uses log base 10. We will make extensive use of dB in frequency spectrum graphs.

Because dB is a power ratio and power dissipated in a load impedance Z is proportional to the square of the voltage across the load or the current through the load, both voltage and current can be expressed in dB. Using the logarithm property from equation (5.2) on page 127,

$$dB = 10\log\left[\frac{P_2}{P_1}\right] = 10\log\left[\frac{V_2^2/Z}{V_1^2/Z}\right] = 10\log\left[\frac{V_2}{V_1}\right]^2 = 20\log\left[\frac{V_2}{V_1}\right] \quad (5.8)$$

Equation (5.8) is equally valid for current ratios: $dB = 20\log[I_2/I_1]$.

For common dB usage in DSP such as *signal to noise ratio* or *total harmonic distortion*, dB can be either positive or negative. If the larger of the two quantities in the ratio is on top, the dB value is positive because the log of values > 1 is positive. If the larger value is on the bottom, the dB value is negative because the log of values between 0 and 1 is negative. The ratio must be positive, as dB is a power ratio and the log function is undefined for values of 0 and less; the absolute value of voltages and currents must be used in the ratio V_2/V_1 and I_2/I_1 .

NOTES:

Decibels of voltage and current ratios divisible by 20 are easy to scale in your head. For example, 120dB is a ratio of 1×10^6 so if V_2 is 10V, V_1 is 10 μ V.

Decibel level	Ratio scale
60dB	1K
40dB	100
20dB	10
0dB	1
-20dB	0.1
-40dB	0.01
-60dB	1×10^{-3}
-80dB	1×10^{-4}
-100dB	1×10^{-5}
-120dB	1×10^{-6}

Table 5.1

Question 5.3: Given a signal S with $1.414\text{V}_{\text{RMS}}$ and noise N of $10\mu\text{V}_{\text{RMS}}$, what is the signal to noise ratio S/N in dB?

NOTES:



Time and Frequency

Periodic Motion

Periodic motion is some action over time which repeats. The time from the start of one repetition to the next is the *time period* or cycle. The classic representation of periodic motion is a pendulum, illustrated in Figure 5.2. We assume that there is some energy source (like a coiled spring) to give the pendulum a slight push on each swing so it does not stop due to friction and gravity. Once the pendulum is swinging, it is easy to see that the motion repeats; it is periodic. A complete period is the time it takes for the center of the disk to start from and return to point *A*. Any other point along the arc *AC* would also work as a start/end point.

To plot the motion of the pendulum we choose time as the x axis and pendulum position as the y axis. To specify position we arbitrarily choose point *A* as $y = 0$, motion from *A* to *C* as positive y values, and motion from *C* to *A* as negative y values. The easiest way to plot this motion is to allow the pendulum to plot its own motion. Suppose a pencil is placed in the center of the pendulum disk and a piece of paper placed underneath moves from the bottom to the top of the diagram as the pendulum moves (like a strip chart recorder). This is shown in Figure 5.3.

NOTES:

Periodic Motion of a Pendulum

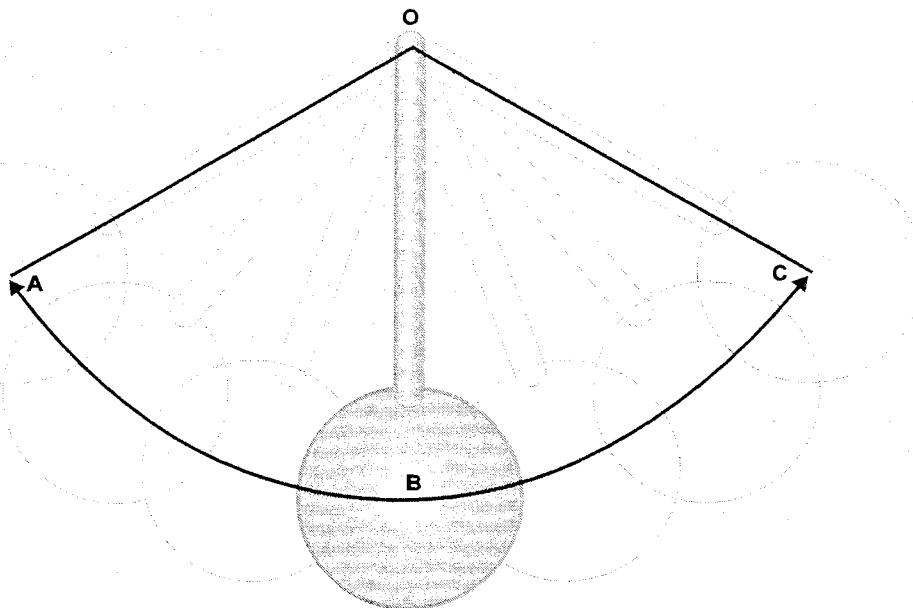


Figure 5.2

The moving paper represents time passing and, as the pendulum swings back and forth, it traces a wave we recognize as a *sinusoid* (or *sine wave*). The origin of the wave depends only on when the pencil starts to draw. If you turn this picture on its side with the pendulum on the right, the x axis represents time increasing to the right and the y axis represents pendulum position (or its amplitude relative to a starting point). Because the pendulum moves past point B in both a positive and negative direction, the times where it crosses B in a positive direction are labelled $B+$, the others $B-$.

NOTES:

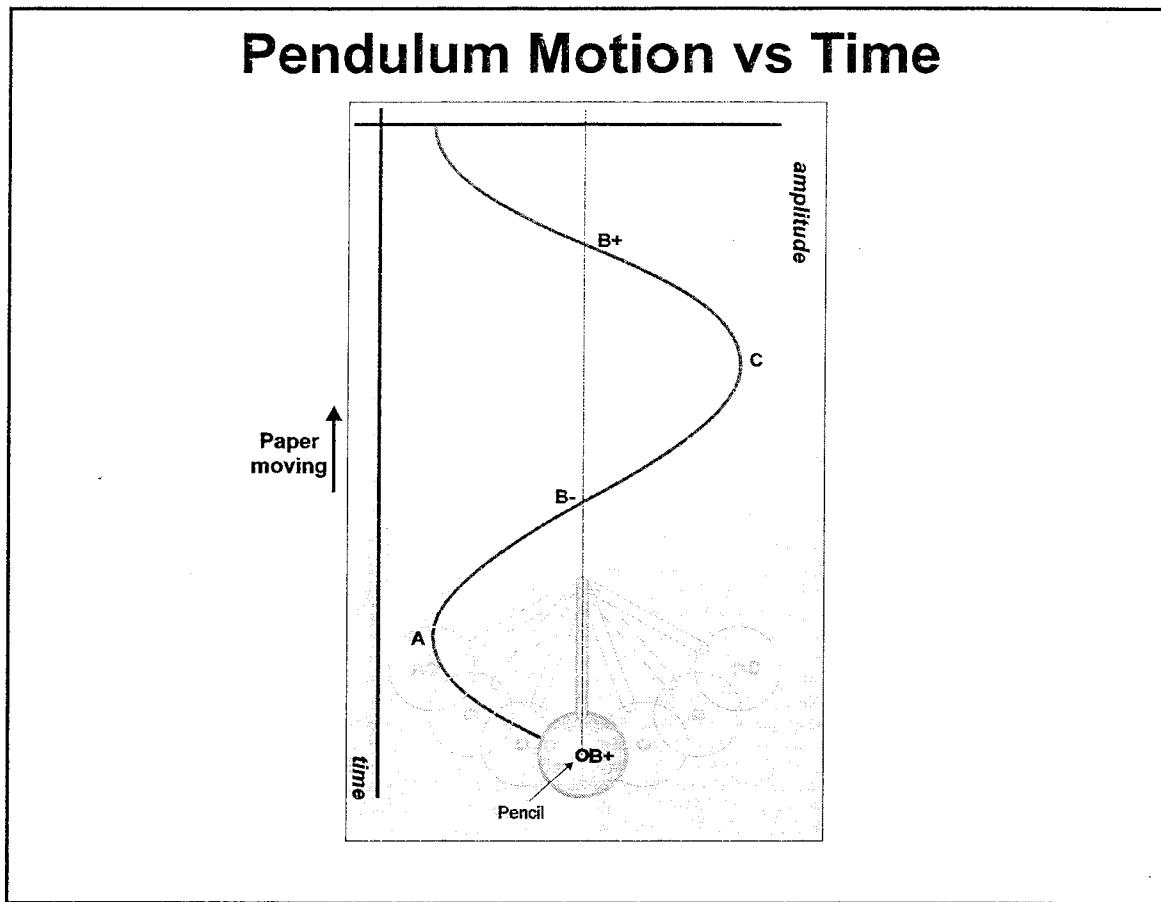


Figure 5.3

Some interesting characteristics can be noted from Figure 5.3—

- ◊ the pendulum is moving fastest at the bottom of its travel (points *B*)
- ◊ the slope of the traced curve is steepest at points *B*
- ◊ the curvature of the traced curve changes at points *B*
- ◊ it is moving slowest at both points *A* and *C*, where it actually stops and reverses direction
- ◊ the slope of the traced curve is least steep at points *A* and *C*, where the slope is zero

NOTES:

- ◊ the faster the pendulum swings, the more times the curve will be traced on a given length of paper (assuming constant paper speed, which must be true if the paper represents time).
- ◊ how frequently the pendulum repeats its motion is its *frequency*. The frequency can be described as the number of crossings of point *B* per unit time, so

$$\text{frequency} = 1 / \text{time}. \quad (5.9)$$

Rotating vector

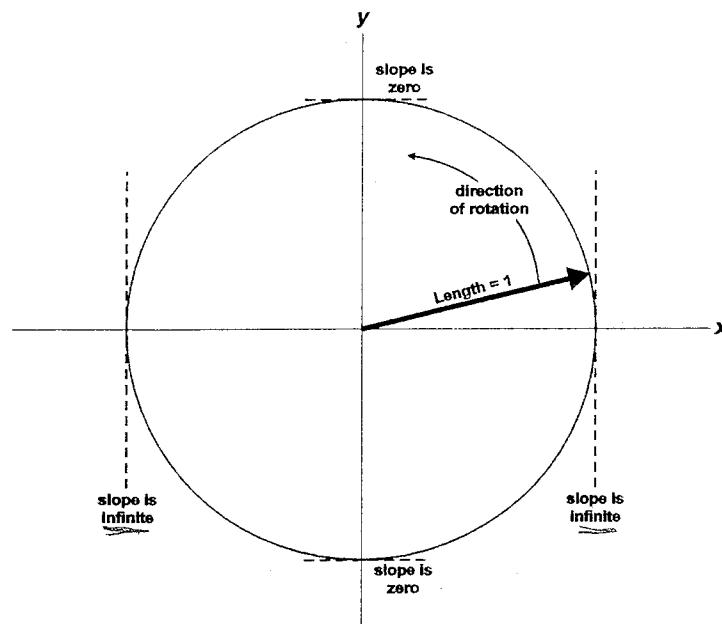
There is another way to represent periodic motion—with a *vector* which rotates about an origin. (*Vector* also has an entirely different meaning in the ATE world; this has no relation to a *test vector*. This vector is a mathematical entity with magnitude and direction.) Consider Figure 5.4, which uses a standard Cartesian coordinate system to show a vector of unit length 1. Time is not represented directly in the figure but is implied by rotation of the vector, i.e. it moves counter-clockwise as time passes. You'll have to use your imagination to pretend that the vector is rotating about the origin like an off-balance propeller or like a hand of a backward clock. How frequently the vector crosses the same point is the *frequency* of its motion. Frequency has units of *Hertz*, which is equivalent to "per second" or "cycles per second."

Now consider the similarity of the motion of a pendulum to the motion of a rotating vector. It helps to look at the horizontal and vertical positions of the vector first and compare them to the pendulum. The rotation of the vector sweeps out a circle of radius *c*, which is shown in Figure 5.5. When the vector (dark arrow) is horizontal and pointing to the right, the slope of the circle at that point (and the slope of the moving end of the vector) is infinite.

Recall that the slope of a line, in this case a line tangent to the right side of the circle, is the change in *y* divided by the change in *x* (or $\Delta y / \Delta x$). As the vector moves a very slight amount in the *y* direction, *x* does not change, thus $\Delta x = 0$, so the slope $= \Delta y / 0 = \infty$. Also notice that the sign of the point is changing from negative to positive. With the vector moving in a positive direction and the sign changing from negative to positive, we can match the point when the vector crosses the *x* axis with points *B+* in Figure 5.3.

NOTES:

Periodic Motion as a Rotating Vector



Time is not visible in this diagram, although it is implied by the rotation of the vector.

Figure 5.4

Next consider when the vector is vertical and pointing up. A small change in x produces virtually no change in y , thus $\Delta y = 0$, so the slope of a tangent to the circle $= 0 / \Delta x = 0$. Also notice that the values of y are changing from increasing to decreasing. This point on the unit circle can be seen as corresponding to point C of Figure 5.3.

If you do this for more points on the circle, you will, in fact, prove to yourself that both the *horizontal* and *vertical* components of the rotating vector will trace out an identical pattern as the pendulum. The vertical component begins at $y = 0$ and increases to $y = 1$, then decreases to $y = 0$ and $y = -1$, then increases back to $y = 0$. This is a *sine* wave. The horizontal component does exactly the same thing, starting at $x = 1$. This is a *cosine* wave. The

NOTES:



conceptual difference is that plotting the rotating vector shows points from two distances (x and y) and y must be plotted separately versus time; the pendulum with moving paper plots y distance versus time directly.

In Figure 5.3, the pendulum moves an equal amount to either side of the center point B . If we specify that the center point is 0 amplitude, the average value of all the points plotted will be zero. In other words, the pendulum spends as much time on one side of the zero point as on the other. On the plot drawn by the pendulum, we can see that the area under the curve drawn to the right of B looks the same as that drawn to the left. Empirically, we see that the “average value” of a sine wave is zero; this is not just an empirical fact; it is a mathematically valid truth as well, and an important concept to remember for later.

Converting angle theta (θ) to frequency and time

Notice the triangle inside the circle in Figure 5.5. In this diagram, it is not just a fixed triangle, but is changing with time as the vector rotates. As the vector goes counterclockwise from the x axis, b gets larger and a gets smaller, while c remains constant at 1. Notice that the ratio of $\frac{b}{c}$ consists of the set of points in the $\sin(\theta) = \frac{b}{c}$ table. The direct result is that *by plotting the values in a sine table versus time, we are plotting the endpoints of a unit length rotating vector which starts at $x = 0$.*

We have a table of $\sin(\theta)$; how can the rotational speed (i.e. the frequency) be specified in the sine equation? Given the angular speed, we can calculate the position by simple multiplication, e.g. $(\text{rotations} / \text{time}) \times (\text{time elapsed}) = \text{rotations}$ where a fractional rotation gives the exact point on the circle irrespective of how many times the vector has rotated. Specifying angular speed as radians per second and time as seconds yields an arc of the circle in radians, i.e.

$$\frac{\text{radians}}{\text{time}} \times \text{time} = \text{radians} \quad (5.10)$$

and allows the vector's position at that time to be calculated exactly. Given the angular measure of ω radians/sec = $2\pi f$, $\sin(\omega t) = \sin(\theta)$. Knowing the angular frequency and the elapsed time since $t = 0$, we can calculate the angle θ . Using θ , with a sine table or calculator we can determine the x and y position of the vector at any time.

What is a radian?

A radian is a way of dividing a circle into a number of parts equal to the length of the circle's radius. In other words, take the radius of the circle and bend it along the circumference of the circle. It will fit exactly 2π times around the circle. This can be verified by the fact that a circle's circumference = πd ($\pi \times \text{diameter}$) = $2\pi r$; the number of “radii” (radians) per circumference thus becomes 2π .

NOTES:

Rotating Vector Creating a Triangle

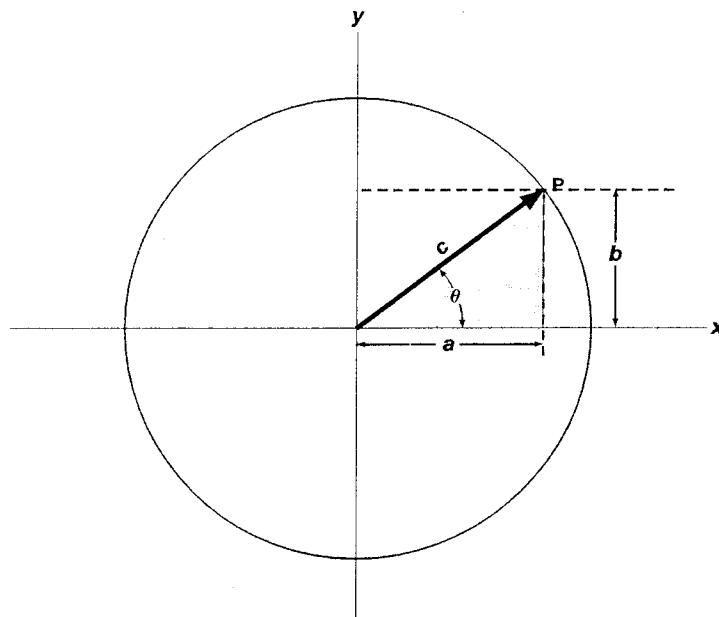


Figure 5.5

You may ask "What happens when the vector has rotated more than one time around during the time interval of measurement?" The sine table takes that into account—it repeats the same values for every rotation. Thus if the angular measurement is $> 2\pi$ radians, divide by 2π , the integer portion of the result is the number of rotations and the remainder is the position during the current rotation.

This subject is one treated casually in many texts, assuming that these results are obvious. It is not obvious to the author and possibly others. Here is a short review:

NOTES:



Mathematically, time becomes part of the sine function by specifying the position θ as angular frequency ω radians/sec multiplied by the elapsed time t in seconds:

$$\theta = \omega \times t \quad (5.11)$$

The equation for the sine of the vector angular frequency becomes more familiar:

$$\sin(\omega t) = \frac{b}{c} = \frac{\text{opposite}}{\text{hypotenuse}} = \sin(\theta) \quad (5.12)$$

One last item involves ω —the frequency in *rotations* per time f (versus *radians* per time) is how often the vector crosses the same point in a given time, and since 1 complete rotation is 2π radians, $\omega = 2\pi f$, giving

$$\sin(\omega t) = \sin(2\pi ft) \quad (5.13)$$

The relationship between time and frequency is one of the most important concepts in DSP. It allows the properties of a time related analog signal to be examined from another perspective to see other information about the content of the signal.

Phase and the cosine function

With a continuously rotating vector, the starting point only depends on our choice for $t = 0$. In other words, the position of the vector at the instant we start looking at it determines the starting point. To use the values from the $\sin(\theta)$ table which start at 0, we chose the positive x axis as the point where $\theta = 0$ and measure angles with respect to that point. What if the vector's rotation starts at another point?

To allow a starting point other than 0, a *phase* can be specified to offset the position. The sine function with a phase offset (or *phase shift*) is written:

$$\sin(\omega t + \phi) \quad (5.14)$$

where ϕ (phi) is the phase offset. Phase offset shifts the vector and its sine wave plot in time with respect to a reference vector. *Without a reference vector, phase has no meaning.* Figure 5.6 shows a vector to point P_2 with a phase difference from the vector to P_1 of angle ϕ . Recall that vector rotation is counter-clockwise, so P_2 vector is ahead, or *leads* P_1 vector by ϕ degrees or radians. If vector P_2 is used as the reference, P_1 is behind, or *lags* P_2 . Phase is normally written in degrees even when radian frequency is used, e.g. $f(t) = \sin(2\pi 1000t + 30^\circ)$.

Looking back at Figure 5.5, the vector was described by using the height of the triangle. The *cosine* of the angle (written $\cos(\theta)$ and defined as $\%_c$) is the equivalent of a sine wave of a vector that starts at $x = 0$. With the sine

NOTES:

function, when the vector is horizontal (angle $\theta = 0$) its height b is zero, so $\sin(0) = 0$. With the cosine function, when the vector is horizontal the triangle's width a is equal to the vector length of 1, so $\cos(0) = \frac{a}{c} = 1$. Choosing a starting value of 1 for cosine places the cosine vector at an angle of $+90^\circ$ with respect to the sine vector when they are plotted on the same graph. When the sine vector crosses the positive x axis the cosine vector crosses the positive y axis. Mathematically,

$$\cos(\theta) = \sin(\theta + 90^\circ) \quad \text{and} \quad \sin(\theta) = \cos(\theta - 90^\circ) \quad (5.15)$$

Phase Difference of 2 Vectors

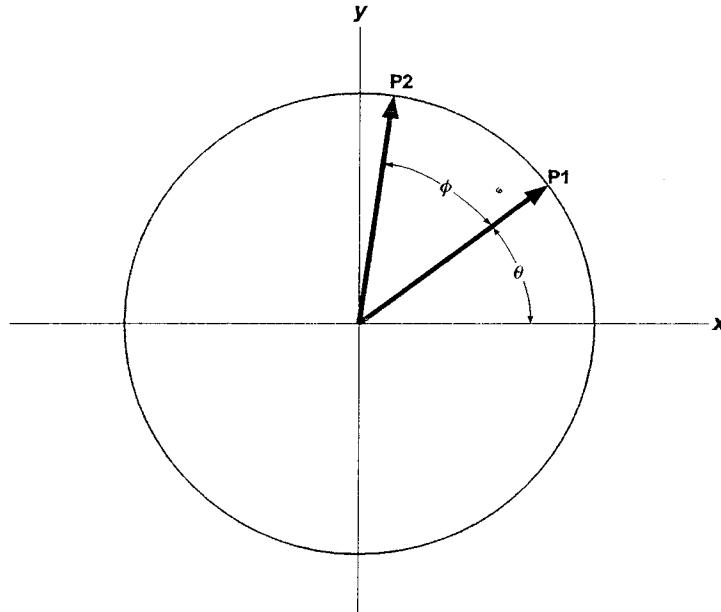


Figure 5.6

NOTES:



RMS value of a sinusoid

RMS is an acronym for *Root Mean Square*, which indicates that a signal has all its components squared, added together, divided by the number of components and the square root of this value taken. RMS is the "DC equivalent" of an AC waveform and indicates how much power the signal will deliver. Mathematically, for a periodic function of time $f(t)$ with period T:

$$RMS = \sqrt{\frac{1}{T} \int_0^T f(t)^2 dt} \quad (5.16)$$

Squaring something means multiplying it times itself, and since negative times negative = positive and positive times positive = positive, squaring makes everything positive, giving a "root power" equivalent that is not zero as the average value will be. Squaring a sine wave is seen in Figure 5.7.

NOTES:

Sine Wave Squared

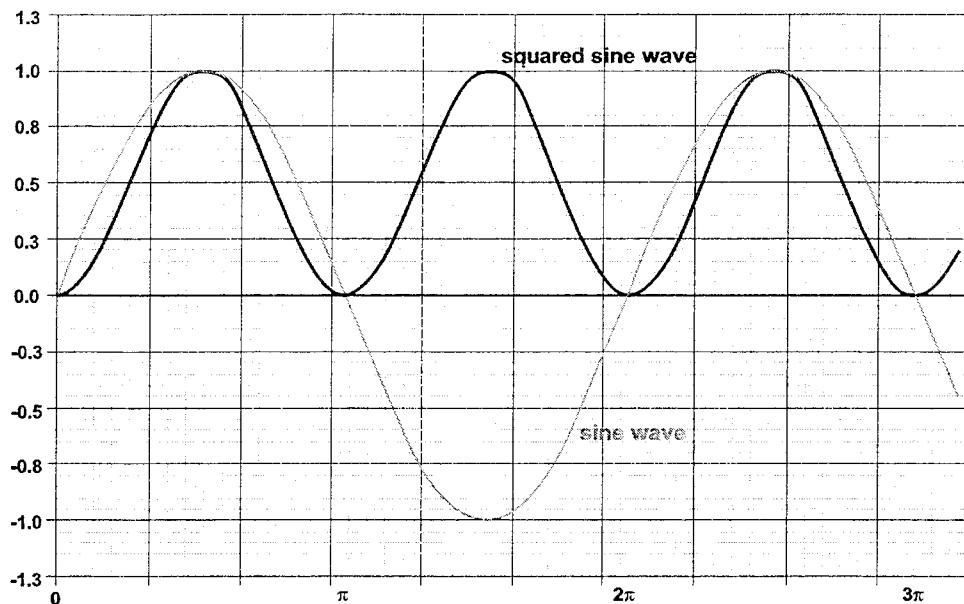


Figure 5.7

The next question is—what is the equivalent power delivered by a single continuous sinusoid? If you could draw a vertical line from the x axis to each point on the squared wave, you could then add all the lines together to make the sum. Notice that all those lines create a solid piece of the waveform which represents the area under the squared wave. How can that area be calculated? The area equals the integral of the waveform from 0 to 1 period T , or

NOTES:

$$Area = \int_0^T [V_{pk} \sin(\omega t)]^2 dt \quad (5.17)$$

which, with a fair amount of detailed calculus or a good reference book² yields the result $V_{pk}^2 / 2$. Taking the square root (the "R" part of RMS) yields an RMS value of

$$V_{RMS} = \frac{V_{pk}}{\sqrt{2}} \quad \text{or} \quad V_{RMS} = 0.707 V_{pk} \quad (5.18)$$

Remember that this simple formula is only true for a sinusoid. A square wave or any other waveform will have an entirely different calculation for RMS value. Fortunately, most of the test parameters we work with require a pure sine wave or a sum of pure sine waves. Also keep in mind that $V_{pk} = V_{pk-pk} / 2$; using one instead of the other will cause errors.

Question 5.4: Given $5\sin(2\pi 1000t + 45^\circ)$, what is **a**) the frequency in Hz? **b**) the phase in degrees? **c**) the phase in radians? **d**) the peak value? **e**) the peak-to-peak value? **f**) the RMS value?

Question 5.5: At $t = 13\mu\text{sec}$, what is the amplitude of the sine wave given in Question 5.4?

Question 5.6: a) Given $v = \cos(2\pi 1000t)$ graphed as a rotating vector, how many times has the rotating vector crossed the positive horizontal axis after 2 seconds? **b)** Which way is the vector pointing at 2 seconds? **c)** At 2.00025 seconds?

NOTES:

Time to frequency translation

Equation (5.9) on page 134 shows how time and frequency are inverse relationships. What exactly does this mean and how can a signal be illustrated in terms of frequency? We have seen a number of amplitude versus time graphs already. A *frequency domain* plot has frequency on the horizontal axis, not time.

Two concepts are important to grasp regarding plots of frequency *spectra* (a range of frequencies):

1. Any given point on the x frequency axis represents the frequency ($f = 1 / \text{period}$) of a single sine wave with amplitude given by the y axis
2. multiple frequencies or a range of frequencies can be plotted, representing a signal that is the sum of a set of sine waves, one for each frequency plotted, with each sine wave's amplitude shown as the height of a frequency plot.

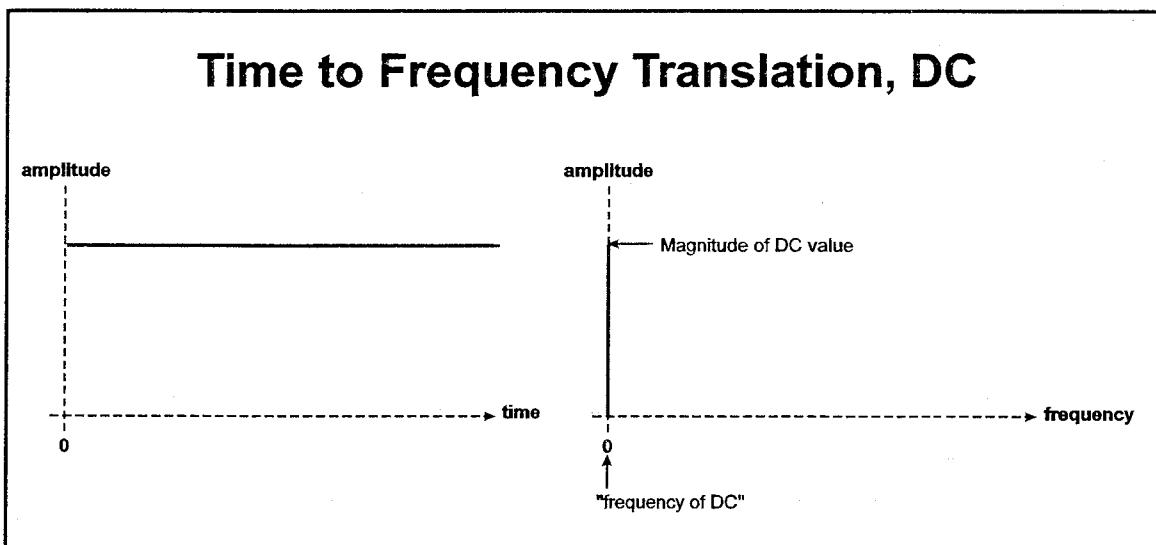


Figure 5.8

Frequency domain plot of DC

DC has no frequency. How can we graph the frequency of a signal with no frequency? How about choosing 0Hz as DC? This is consistent with the results of an FFT displayed with frequency bins (explained in chapter 6) and with frequency plotted on a linear horizontal axis. If the frequency axis is logarithmically scaled, it is not possible to display DC because a log scale cannot show the log of 0Hz.

NOTES:

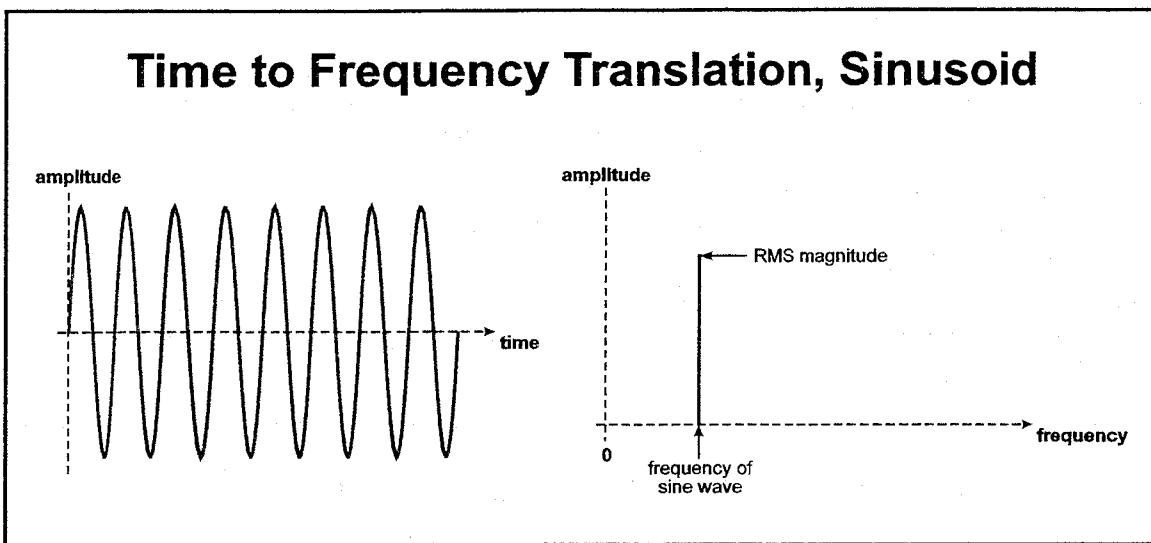


Figure 5.9

Frequency domain plot of a sine wave

A sine wave has, as we will learn soon, a single “pure” frequency which can be seen on a frequency plot as a single vertical line. The height of the line shows the amplitude of the sine wave *as an RMS quantity*. What does RMS mean and why not just plot the maximum amplitude or the average value? Recall that the average value of a sine wave is zero; we certainly can’t plot the average value to show any useful information. If we need to plot the frequency spectrum of a complex signal such as voice (which is not just a simple sine wave) the maximum amplitude may be a single large peak, with most of the signal having much less amplitude. Neither of these accurately represents a signal. What we need to know is how much power is delivered to a load by a signal, or more fundamentally, how much energy can be delivered by the signal in a given time. This is the RMS value as discussed earlier and is given by the height of the amplitude spike of each sine component on the graph.

NOTES:

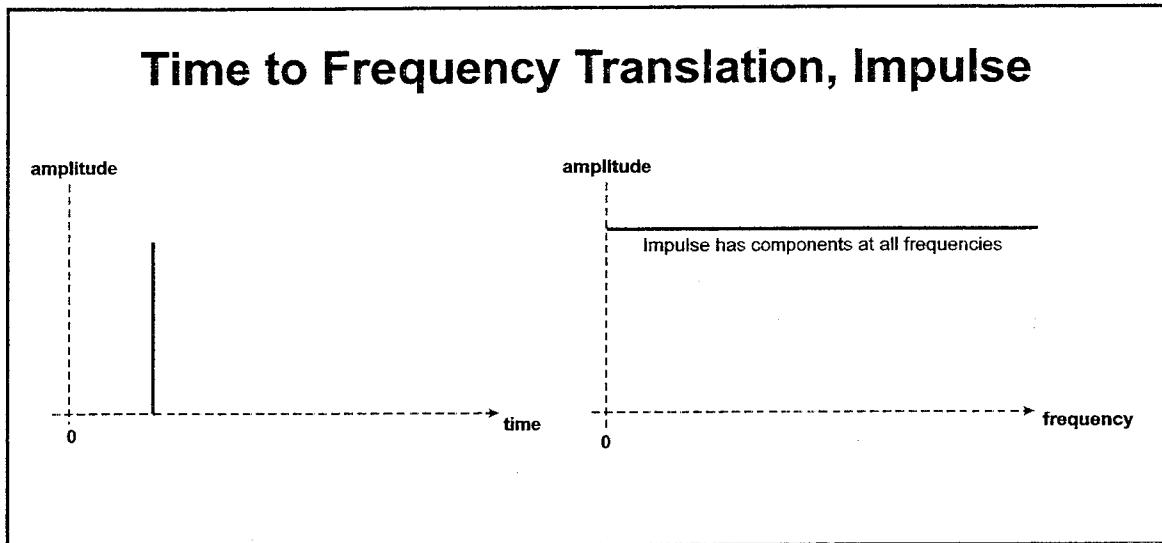


Figure 5.10

Frequency domain plot of an impulse

DC is a horizontal line in time and represents the same value for an infinitely long time. Its frequency can be described as $\frac{1}{T_0} = 0$. An impulse is the opposite and represents a finite amplitude value for a single, infinitely short time; its frequency can be described as $\frac{1}{T_0} = \infty$. An impulse in time contains all frequencies from DC to light; naturally, physical constraints prevent the existence of a true impulse, but its characteristics are important to understand when dealing with instantaneous changes in signal value. An impulse with amplitude 1 is sometimes called the *Dirac delta function*.

Question 5.7: A sinusoidal signal is often called “alternating current” or “AC.” What is the most common source of alternating current?

NOTES:

Fourier series

The previous section alluded to summing sine waves in the discussion of frequency spectra on page 143. It is possible to create a complex waveform by summing pure sine waves with different amplitudes and frequencies and to decompose a complex signal into a sum of sinusoids of different amplitudes and frequencies. The summation is known as a *Fourier series*, named after Joseph Fourier, who did groundbreaking work on the analysis of heat conduction³ and applied harmonic summation techniques to a general problem.

Why is the Fourier series important? The Fourier series and its counterpart the Fourier transform allow virtually any real world signal to be decomposed into a collection of sine waves. Arbitrary signals are complicated and difficult to analyze mathematically. Sine waves are easy to analyze mathematically. By using the Fourier equivalent of a real signal, each of its sinusoidal components can easily be analyzed, with the knowledge that the analysis results will apply to the original waveform.

Fourier Conditions

Not every mathematical function of a waveform can be created or analyzed using a Fourier series. The conditions a signal must meet so that it can be decomposed into a Fourier series are described below.

1. the signal (or, more formally, the function) is single valued everywhere, i.e. one and only one y point corresponds to each x point
2. the signal is periodic, i.e. it repeats itself at a fixed interval
3. the area bounded by the signal over one period is finite

There are a couple of additional conditions regarding "discontinuities" and "finite maxima and minima," but the signals we encounter (and most real world electrical signals) will meet all the above conditions.

Calculating the sine and cosine components

To create a complex signal that meets the Dirichlet conditions, it must be represented by a sum of sinusoids as given in the following function:⁴

$$f(t) = a_0 + A[a_1 \cos(\omega_1 t) + a_2 \cos(2\omega_1 t) + \dots + b_1 \sin(\omega_1 t) + b_2 \sin(2\omega_1 t) + \dots] \quad (5.19)$$

or, using a summation,

$$f(t) = a_0 + A \left[\sum_{n=1}^{\infty} [a_n \cos(n\omega_1 t) + b_n \sin(n\omega_1 t)] \right] \quad (5.20)$$

NOTES:



where:

- ◊ a_0 is the DC component
- ◊ A is an overall scale factor for all harmonic components
- ◊ ω_1 is the frequency of harmonic 1 (the *fundamental frequency*)
- ◊ n is an integer multiplier of the fundamental frequency for each harmonic term
- ◊ ϕ is a phase shift available for any harmonic.

This tells us that not only can we sum a series of sine and cosine waves to create any other wave, but also that the frequencies of the sinusoids are integer multiples (called *harmonics*) of a single fundamental frequency. The big hurdle is in calculating the a_n and b_n constant coefficients. This is done by using boundary conditions on a set of integral equations related to the sine and cosine and putting them into equation (5.20). We do not need to derive the technique here, as it can be found in Hayt and Kemmerly⁵, Wylie⁶ and other college mathematics and engineering texts. The important thing is to understand that any signal meeting the Dirichlet conditions can be created or analyzed as a sum of pure sine and cosine waves. Fourier equations for many common waves have been analyzed and are available in various engineering and mathematical references and text books.

Three common waveforms are given in the following equations:

$$\text{Square wave: } f(t) = \frac{4V_m}{\pi} \left[\sum_{n=1,3,5,\dots}^{\infty} \frac{\sin(n\omega_1\pi t)}{n} \right] \quad (5.21)$$

$$\text{Triangle wave} = \frac{8V_m}{\pi^2} \left[\sum_{n=1,3,5,7,\dots}^{\infty} \left((-1)^{\frac{n-1}{2}} \right) \frac{\sin(n\omega_1\pi t)}{n} \right] \quad (5.22)$$

$$\text{Half Sine wave: } f(t) = 1 + \frac{V_m}{\pi} \left[\frac{\pi}{2} \cos(5\pi\omega_1 t) - \sum_{n=2,4,6,8,\dots}^{\infty} \left((-1)^{\frac{n}{2}} \right) \left(\frac{2}{(n^2-1)\pi} \cos(5n\pi\omega_1 t) \right) \right] \quad (5.23)$$

Notice the correspondence of terms in these equations with equation (5.20), e.g. a_0 for the square and triangle waves is zero and A for the square wave is $4V_m / \pi$. Symmetry of a given waveform about the vertical axis (at $t = 0$) or about the horizontal axis means some terms evaluate to zero, making the equations simpler. Notice that the above equations contain only sine or cosine terms, not both. Also notice that the square wave has only positive odd harmonics, the triangle wave has alternating positive and negative odd harmonics, and the half sine wave has a DC term, a 1st fundamental (odd) harmonic and alternating positive and negative even harmonics starting at the 2nd. (Be aware that other equations for square, triangle and half sine waves are possible.)

NOTES:

Lab Exercise 5.1—Creating and examining a Fourier series

Lab goals

- ❖ Create and view a sine wave by making a Fourier series which has only 1 term
- ❖ Create and view a square wave by having the lab use equation (5.21) to create all harmonics up to 20th that are required for the square wave. Each harmonic is plotted individually in a different color, and the sum of all harmonics is plotted in black.

Lab objectives

- ❖ Get acquainted with the laboratory software
- ❖ Illustrate the phase relationship between sine and cosine
- ❖ Show the correlation between the DC offset of a wave and its 0th harmonic term
- ❖ Illustrate that a complex wave can be created with a sum of simple sine waves at integer frequency multiples of the fundamental

This lab allows you to create your own Fourier waveform by entering coefficients for each harmonic term in the general Fourier series as given by equation (5.20) on page 146. It also allows you to view waves which are created as the sum of sine and cosine terms that are precalculated. First, we must become familiar with the software. Start the *DSP Lab* software and press the *Fourier Series* button.

Special note: The lab software does not fully implement equation (5.20). Each harmonic can contain only a sine or cosine component, not both. This allows most common waveforms to be created while eliminating the complexity of entering 2 amplitudes and phases for each harmonic term.

NOTES:



Fourier Waveform

Refer to equation (5.20) on page 146 and note the following correlations between constants in the equation and parameters in the software and the purpose of the controls in the software:

Equation (5.20)	DSP Lab	Purpose
ω_1	Fundamental Frequency edit box	Sets the frequency of the fundamental term (Harmonic Number 1).
A	Overall Scale Factor	Allows all terms in Fourier sum to be multiplied by a constant factor. Necessary to set peak value of Fourier sum to a specific level.
n	Harmonic Number 0, 1, 2... edit box	Number of the harmonic term to be set or changed, where 0 is DC, 1 is the fundamental, 2 is the 2nd harmonic, etc. Maximum is 20th harmonic.
a_n, b_n	Peak Amplitude	Sets maximum amplitude (V_{pk}) of each cosine or sine term by setting the coefficient.
sin or cos	Term Type	Sets whether the selected harmonic is a cosine or sine term.
ϕ	Harmonic Phase	Allows a phase shift to be added to any harmonic term.
	Plot Color	Allows each harmonic term to be plotted on oscilloscope in a different color. The sum of harmonics is always plotted in black.
	Plot	When checked, the selected harmonic term is plotted individually on the oscilloscope. Must be set for each harmonic to be plotted; the sum of harmonics is always plotted

Equations for a square (Equation (5.21)), triangle (Equation (5.22)) and half wave rectified sine wave (Equation (5.23)) are programmed into the buttons in the *Preset Waveforms* panel. When you press one of the buttons, the appropriate *Overall Scale Factor*, *Peak Amplitude*, *Term Type* and *Harmonic Phase* are set for each harmonic out to the 20th.

NOTES:



Oscilloscope

There is also an *Oscilloscope* which shows the waveforms created in the *Fourier Series Waveform Generator*. It has amplitude and time scaling knobs just like a real oscilloscope and they work the same as a real oscilloscope, except they set the full scale value not the “per division” value. Notice that when the mouse cursor passes over the oscilloscope graph, the amplitude and time are displayed in the status panel at the bottom of the window. This is how you get measurement points from the oscilloscope. Because the mouse resolution is only as good as the number of pixels on the screen, there is round-off error in the status panel values, so they are only approximate. The most accurate values are achieved by expanding the *Amplitude* and *Time Scale* to the maximum that still shows the desired point in the graph before you take a measurement.

Becoming familiar with the Fourier Waveform lab

In the *Fourier Series Waveform Generator*, set these values:

1. Press *Clear Harmonics* to reset all values and remove any harmonic data
2. Set *Fundamental Frequency* = 1000
3. Set *Overall Scale Factor* = 1
4. Select *Harmonic Number 0* and set *Peak Amplitude* = 0.0
5. Select *Harmonic Number 1* and set *Peak Amplitude* = 1.0, *Term Type* = Sine and *Harmonic Phase* = 0
6. Change to the *Oscilloscope* by clicking the tab at the top
7. Set *Amplitude* knobs to 2V
8. Set *Time Scale* knobs to 2msec

Using the cursor over the oscilloscope to see the time and amplitude information in the status bar at the bottom of the window, answer these questions:

Lab Question 5.1.1: a) What is the period of the waveform? b) What is the peak amplitude? c) What is the waveform type? d) What is the DC offset? e) What is the value of the wave at $t = 0$?

Change to the *Fourier Series Waveform Generator*, set *Fundamental Frequency* = 2000, change to the *Oscilloscope*, set *Time Scale* = 1msec.

NOTES:



Lab Question 5.1.2: What is the period of the waveform?

Change to the *Fourier Series Waveform Generator*, set *Overall Scale Factor* = 0.5, change to the *Oscilloscope*, set the *Amplitude full scale* knob to 1V

Lab Question 5.1.3: What is the peak amplitude of the waveform?

Change to the *Fourier Series Waveform Generator*, select *Harmonic Number 0* and set *Peak Amplitude* = 0.8 and change to the *Oscilloscope*.

Lab Question 5.1.4: What is the DC offset of the waveform?

Change to the *Fourier Series Waveform Generator*, select *Harmonic Number 1* and set *Peak Amplitude* = 0.3, change to the *Oscilloscope*

Lab Question 5.1.5: What is the peak amplitude of the waveform (ignoring the DC offset)?

Examine sine, cosine and phase difference

In the *Fourier Series Waveform Generator*, set these values:

1. Press *Clear Harmonics* to reset all values and remove any harmonic data
2. Set *Fundamental Frequency* = 1000
3. Set *Overall Scale Factor* = 1
4. Select *Harmonic Number 0* and set *Peak Amplitude* = 0.0
5. Select *Harmonic Number 1* and set *Peak Amplitude* = 1.0, *Term Type* = Sine and *Harmonic Phase* = 0
6. Change to the *Oscilloscope* by clicking the tab at the top or pressing Ctrl-Tab
7. Set *Amplitude* knobs to 2V
8. Set *Time Scale* knobs to 2msec

Again use the oscilloscope cursor to read values from the graph.

NOTES:

Lab Question 5.1.6: a) What is the period of the waveform? b) What is the peak amplitude? c) What is the waveform type? d) What is the DC offset? e) What is the value of the wave at $t = 0$?

Change to the *Fourier Series Waveform Generator*, select *Harmonic Number 1* and set *Term Type = Cosine*, change to the *Oscilloscope*

Lab Question 5.1.7: a) What is the period of the waveform? b) What is the peak amplitude? c) What is the waveform type? d) What is the DC offset? e) What is the value of the wave at $t = 0$?

Change to the *Fourier Series Waveform Generator*, select *Harmonic Number 1* and set *Term Type = Sine*, set *Harmonic Phase = 90* (this is in degrees, not radians), change to the *Oscilloscope*

Lab Question 5.1.8: a) What is the period of the waveform? b) What is the peak amplitude? c) What is the waveform type? d) What is the DC offset? e) What is the value of the wave at $t = 0$?

Lab Question 5.1.9: a) Is it possible to distinguish between a cosine wave with 0° phase shift and a sine wave with 90° phase shift? b) Is the following a valid statement?

If any complex signal can be analyzed as the sum of a Fourier series of sine and cosine waves and a cosine wave is the same as a sine wave with a 90° phase shift, then any signal can be analyzed as a Fourier series of sine waves.

NOTES:



Examine a square wave created with a Fourier series

Press the Calculator button (bottom right of window) and set its *View* to *Scientific*.

In the *Fourier Series Waveform Generator*, make sure the *Fundamental Frequency* is set to 1000 then press the *Square Wave* button. Recall the square wave equation (5.21) on page 147, repeated here:

$$f(t) = \frac{4V_m}{\pi} \left[\sum_{n=1, 3, 5, \dots}^{\infty} \frac{\sin(n\omega_1\pi t)}{n} \right]$$

Lab Question 5.1.10: Look at the square wave equation and, with $V_m = 1$, calculate the overall scale factor that every harmonic is multiplied by ($4V_m / \pi$).

Lab Question 5.1.11: How does this number compare to the *Overall Scale Factor* in the *Fourier Waveform* tab?

Click the up arrow in the *Harmonic Number* edit box to cycle through all harmonics up to 20.

Lab Question 5.1.12: Which harmonics are set? Compare this to the $n = 1, 3, 5, \dots$ summation condition in equation (5.21). Also notice that set harmonics have a *Plot Color* and *Plot* is checked.

Set the *Harmonic Number* to 3 by typing it in or by using the arrows.

Lab Question 5.1.13: What is the *Peak Amplitude* shown? Notice that, for $n = 3$, the peak value of the 3rd harmonic term $\sin(n\omega_1\pi t) / n = 1/3$ at its peak. Repeat this exercise for *Harmonic Number* = 5 and 7.

Change to the *Oscilloscope* tab. Make sure *Amplitude* is set to 2V and *Time Scale* is set to 2msec. Do you see a black square wave? It is created by the summation of all the other sine wave harmonics plotted.

NOTES:

Lab Question 5.1.14: **a)** Which harmonic term has an amplitude larger than the square wave? **b)** What is the frequency of this term versus the square wave frequency?

Adjust the *Amplitude* and *Time Scale* knobs so you can better see the smaller valued harmonics (e.g. try 500mV and 200 μ sec). Notice the relative amplitudes and frequencies of the harmonics. Notice how all the harmonic terms cross the time axis at the point where the fundamental crosses; this is the fast moving part of the square wave.

NOTES:



Complex numbers

The j operator

In the 1700s, Swiss mathematician Leonhard Euler (considered by many to be the world's greatest mathematician⁷) used the abstract concept of $\sqrt{-1}$ to solve 2nd order differential equations. The number cannot exist, but it has properties which make it a pragmatic mathematical tool. In practice, an imaginary number is a real number multiplied by $\sqrt{-1}$, represented in engineering by the symbol j .

By combining real and imaginary numbers into *complex numbers*, Euler devised a way to represent a set of numbers which are a superset of the real numbers we use normally. The properties of complex numbers are consistent with those of real numbers and can be used in combination with them as an analytical tool. The set of complex numbers is used extensively in DSP to convert time samples into frequency data.

A complex number consists of a *real* and an *imaginary* part. The imaginary part is created by multiplying a real number by the j operator. Combining a real and imaginary component results in a complex number as follows:

$$\text{complex} = Re + jIm \quad (5.24)$$

The complex plane

The complex plane has a real and imaginary axis and is used to graphically represent the complex number set. It is the complex number equivalent of using a one dimensional number line to represent the real number set. Although a complex number graph looks very similar to the rotating vector in Figure 5.5 on page 137 and uses the same trigonometry to calculate a vector length, the complex plane shows nothing about time nor does it have any inferred "rotation" of the vector representing the number's value.

A complex number is represented by a vector that is the hypotenuse of a triangle. The complex plane plots the real term along the horizontal axis and the imaginary term along the vertical axis. In Figure 5.11, with respect to the x axis, multiplying a point by -1 is equal to a phase shift of 180°. By using $\sqrt{-1}$ (or j) as a 90° phase shift, the "imaginary" numbers are plotted along the vertical ji axis.

As with any vector, complex numbers can be represented in both rectangular and polar form. The rectangular form is especially useful for "real-world" math operations on a computer because addition, subtraction, multiplication, etc. of complex quantities can be done as separate operations on each real and imaginary component. Designating a point in the complex plane in rectangular form involves the usual trigonometry to navigate in the horizontal and vertical directions. It can be seen from Figure 5.11 that

$$a = r\cos\theta \quad \text{and} \quad b = r\sin\theta \quad (5.25)$$

NOTES:

Vector Representing a Single Number in the Complex Plane

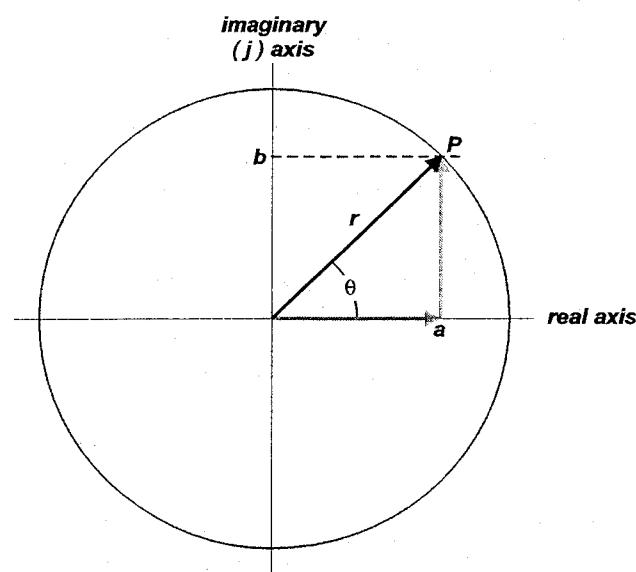


Figure 5.11

To specify point P in terms of real and imaginary parts of vector r , the real component is $r\cos\theta$ and the imaginary component is $r\sin\theta$, yielding a complex value of

$$P = r(\cos\theta + j\sin\theta) \quad (5.26)$$

implying that any arbitrary complex value can be represented as

$$a + jb = r(\cos\theta + j\sin\theta) \quad (5.27)$$

NOTES:

The “+” sign can lead to confusion in equation (5.27)—it is important to note that “ $a + jb$ ” represents a single complex value, *not* an addition operation.

Euler's Relationship

It can be shown⁸ that, for e raised to a complex power ($x + iy$), the following is true:

$$e^{x+jy} = e^x (\cos y + j \sin y) \quad (5.28)$$

Using a standard property of exponents $x^u + v = x^u x^v$, the following is also true:

$$e^x e^{jy} = e^x (\cos y + j \sin y) \quad (5.29)$$

Substituting the more traditional θ for y and dividing both sides by e^x gives us:

$$e^{j\theta} = \cos \theta + j \sin \theta \quad (5.30)$$

If both sides of equation (5.30) are multiplied by a real value r , the following relationship is established:

$$r e^{j\theta} = r(\cos \theta + j \sin \theta) \quad (5.31)$$

This is known as Euler's relationship; comparing this with equation (5.26) shows that any complex value P can be represented with a complex exponential function:

$$P = r e^{j\theta} \quad (5.32)$$

Equation (5.32) is the polar form of a complex value. Euler's relationship, together with the Fourier integral transform of a continuous function in time, given as

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \quad (5.33)$$

are the basis for frequency analysis of electrical signals. Equation (5.33) translates any continuous function of t into a continuous function of f , with ω being the traditional $2\pi f$. Equation (5.33) is the origin of customary frequency plots such as the filter frequency response shown in Figure 2.2 on page 53.

NOTES:

Conversion between polar and rectangular

Pythagoras's triangle relationship can be used to solve for the magnitude and phase of a complex number the same as a vector. Given a complex number $a + jb$, the following relationships show the calculation for magnitude and angle in the complex plane, from rectangular to polar:

$$\text{magnitude} = \sqrt{a^2 + b^2} \quad \text{Phase} = \text{atan}\left(\frac{b}{a}\right) \quad (5.34)$$

Given the vector magnitude r and the phase angle θ from the positive real axis as $r e^{j\theta}$, the following relationships show the conversion from polar to rectangular:

$$a = r \cos \theta \quad b = r \sin \theta \quad (5.35)$$

Why would you ever need these relationships in DSP? When you digitize a signal and send a set of time samples to the array processor, it may return the rectangular form of a complex array as 2 arrays of real numbers, the real and imaginary values. To plot a frequency spectrum (called a Bode plot), you must convert these numbers into magnitude and phase information using the above equations.

Question 5.8: a) What is the magnitude and phase (in degrees) of $3 + j4$? b) What is the rectangular form of the complex quantity $9e^{j\pi/2}$?

Representing a real sine wave in the complex plane

Equation (5.31) can be solved for $\cos \theta$ and $\sin \theta$, which are both functions of the real variable θ . Replacing θ with ωt as discussed in *Converting angle theta (q) to frequency and time* on page 136 results in the following:

$$\cos \omega t = \frac{e^{j\omega t} + e^{-j\omega t}}{2} \quad \text{and} \quad \sin \omega t = \frac{e^{j\omega t} - e^{-j\omega t}}{2j}. \quad (5.36)$$

From equation (5.31) with $r = 1$, it can be seen that at any point in time, $e^{j\omega t}$ and $e^{-j\omega t}$ are each a complex number. Representing a real sinusoid in the complex plane thus involves 2 complex quantities changing with time. For an interesting treatment of this subject, see page 462 of reference 7 listed at the end of this chapter.

NOTES:



Key Points of This Chapter

- ❖ A logarithm is a transform which relates exponents to a base. To make calculations using logarithms, it is often necessary to take an anti-log to translate log values back to standard values. Logarithm values are stored in a precalculated table or a calculator.
- ❖ Logarithms allow exponential ranges of values to be graphed or plotted on a linear scale. Log plots represent small values over a larger range and large values over a smaller range.
- ❖ Decibels (dB) is a logarithmic power ratio given by equation (5.8) on page 129. The ratio can be referenced to a known (measured) value or a common denominator (e.g. 1mW is the reference denominator for dBm).
- ❖ Decibels are useful when graphing or comparing signal values that cover a wide range, e.g. diode current versus voltage or a signal that is large relative to circuit noise.
- ❖ There is a specific relationship, for periodic mathematical functions and for periodic electrical signals, between the time of a period and the frequency of occurrence of the period.
- ❖ The time and frequency information can be calculated and graphed.
- ❖ Virtually all periodic real world electrical signals can be decomposed into a series of sine and cosine components with integer related frequencies. It is done with Fourier analysis and the result is called a Fourier series.
- ❖ Fourier analysis allows complicated signals to be analyzed as a sum of simple sine waves. It is a way to calculate the Fourier components of a waveform so they can be plotted as sinusoidal components versus frequency.
- ❖ Complex numbers give us the mathematical tools to calculate frequency information about a signal.
- ❖ The rectangular form of complex numbers is useful for computerized analysis of waveforms.

NOTES:

Answers to chapter questions

Answer 5.1: 12

Answer 5.2: $68 * 777 = e^{[\ln(68 * 777)]} = e^{[\ln(68) + \ln(777)]} = e^{[4.2195 + 6.6554]} = 52836$

Answer 5.3: 103dB

Answer 5.4:

- a) 1000Hz
- b) 45°
- c) $\pi/4$
- d) 5
- d) 10
- e) 3.536 (or $5/\sqrt{2}$)

Answer 5.5: 3.812

Answer 5.6: a) 2000 b) vertical up c) horizontal left

Answer 5.7: A wall power outlet.

Answer 5.8: a) $5\angle 53.13^\circ$ b) $0 + j9$

NOTES:

Answers to Lab Exercise 5.1

5.1.1: a) 1msec b) 1V c) sine wave d) 0V e) 0V

5.1.2: 0.5msec

5.1.3: 0.5V

5.1.4: 0.8V

5.1.5: 0.15V

5.1.6: a) 1msec b) 1V c) sine wave d) 0V e) 0V

5.1.7: a) 1msec b) 1V c) cosine wave d) 0V e) 1V

5.1.8: a) 1msec b) 1V c) sine wave d) 0V e) 1V

5.1.9: a) No b) Yes

5.1.10: 1.273239544735

5.1.11: It's the same

5.1.12: The odd harmonics have an amplitude value set

5.1.13: $0.3333333333333333 \left(\frac{1}{3}\right)$, $0.2 \left(\frac{1}{5}\right)$, $0.142857142857143 \left(\frac{1}{7}\right)$

5.1.14: a) The fundamental b) the same frequency

NOTES:

References

1. Morris Slutzberg and William Osterheld, *Essentials of Radio*, McGraw-Hill Book Company Inc., 1948, pg 324.
2. Samuel M. Selby Ph.D, Sc.D., *CRC Standard Mathematical Tables*, Twenty-first Edition, 1973, The Chemical Rubber Company.
3. James D. Broesch, *Digital Signal Processing Demystified*, HighText Publications, 1997, pg 75.
4. Ibid.
5. Ibid.
6. Ref. 2, pp 214-256.
7. Richard G. Lyons, *Understanding Digital Signal Processing*, Addison Wesley Longman, Inc., 1997, pg 460.
8. Ref. 6, pg 757-759.

NOTES:

Sampling

Goals

- ◊ Explain how to create a signal with samples
- ◊ Explain how to sample a continuous signal
- ◊ Discuss the Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) algorithms
- ◊ Describe the results of processing a sample set via DFT/FFT
- ◊ Explain coherent sampling in detail
- ◊ Discuss the Inverse DFT/FFT and its use to create a sample set for a complex waveform

Objectives

- ◊ To understand the principles of sampling, especially coherent sampling
- ◊ To be able to apply sampling principles in a test situation to generate an analog signal with a waveform generator or DAC
- ◊ To be able to apply sampling principles in a test situation to digitize an analog signal with a waveform digitizer or ADC

What you will learn

- ❖ How many samples are required to retain all necessary signal information
- ❖ What are the sample period and frequency and test signal period and frequency
- ❖ The DFT algorithm
- ❖ The FFT algorithm
- ❖ The advantage and limitation of FFT versus DFT
- ❖ How to use to a DFT/FFT algorithm
- ❖ How a single set of time samples represents more than one set of frequency samples (signal replication)
- ❖ How replicated signals can interfere with frequency data from an DFT/FFT (aliasing)
- ❖ How to prevent aliasing with a filter
- ❖ How an incomplete or non-coherent sample set can cause $\sin(x)/x$ amplitude error when generating a waveform
- ❖ How an incomplete or non-coherent sample set can cause false distortion data (leakage) when digitizing a waveform
- ❖ Use of mathematical algorithms (window functions) to minimize leakage
- ❖ The benefits of coherent sampling and how it is done
- ❖ The relationship between sample frequency, test signal frequency, how many test signal periods are sampled and how many samples are taken
- ❖ Coherent sampling requirements for the number of samples versus the number of test period cycles
- ❖ How to use the coherent sampling equation to achieve a desired test frequency
- ❖ Calculation of the frequency axis on DFT/FFT results from a coherent sample set (Fourier Frequency FF, Unit Test Period UTP, frequency bin, frequency resolution, maximum frequency)
- ❖ How to create the discrete samples needed to generate a signal
- ❖ How to digitize (collect samples from) a continuous signal

NOTES:

Requirements for sampling

Consider a continuous time signal that must be stored as a sequence of digital numbers. The way this is done is, at certain intervals, the value of the waveform is converted to a digital number. These digital numbers are stored as "sample points" for the waveform. If the samples are taken in a certain way, an infinite valued signal can be completely represented by a set of finite sample points.

The interval between samples is called the *sample time*, abbreviated t_s . The time must be the same between each sample, otherwise the mathematics to analyze the samples are virtually impossible. The *sample frequency* F_s is the inverse of the sample time, i.e. $F_s = 1/t_s$. With a continuous signal, there is only one frequency of concern, the *test signal frequency*, F_t . With a sampled data system there are two frequencies to keep track of, the test frequency and the sample frequency, and they have certain requirements that must be met.

Shannon's theorem

In 1949, C. E. Shannon¹ proved that a continuous mathematical function can be completely determined by samples taken at twice the highest frequency component in the function. This applies to classic mathematical functions that go from minus to plus infinity. A small set of samples of a waveform do not cover such a broad range, however, and have a slightly more restrictive sample requirement to enable complete information recovery. When a periodic signal such as a sine wave is sampled at exactly $\frac{1}{2}$ the signal frequency, the information is not quite enough to later reconstruct the signal, as seen in Figure 6.1.

Nyquist's limit

Harry Nyquist, demonstrated that, to be able to recreate a signal from its samples, *you must sample at a rate higher than 2 times the highest frequency of interest contained in the signal*. The important point is that more than 2 samples per test signal period are required; not much more, "just one extra sample will do the trick."² Naturally, the more samples per period, the more information that can be gleaned from the sample set.

The phrase *highest frequency of interest* relates to a signal which can be decomposed into a Fourier sum and has some very interesting implications. It means that the test signal being sampled can have frequency components which are not in the range of interest and the signal of interest can be extracted anyway. Said another way, different frequency signals can be summed and, by sampling the summed signal with different sample rates, the different signals can be extracted separately. This is called *bandpass* or *sub-Nyquist* sampling. This type of sampling is used mostly in communications circuits such as cellular telephones or satellites and will not be discussed further in this course.

NOTES:

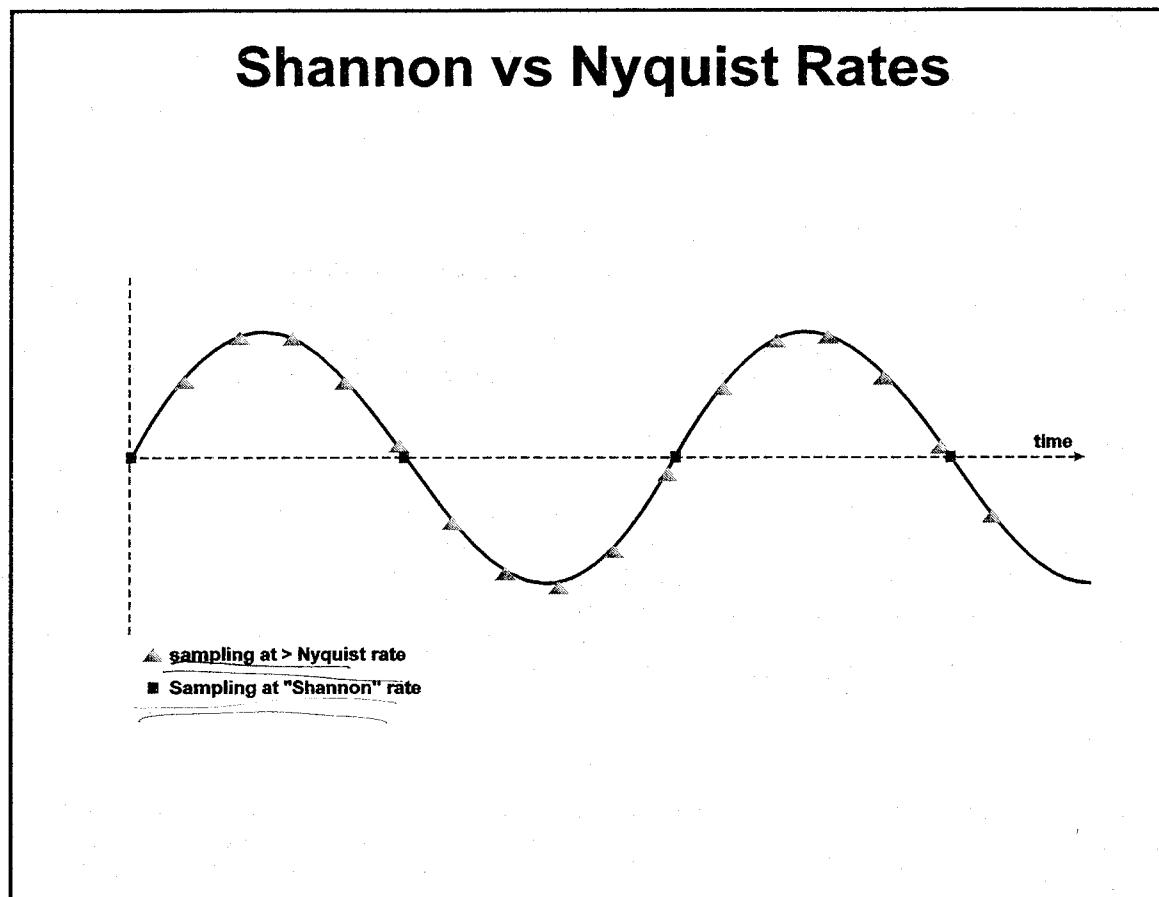


Figure 6.1

Question 6.1: What are 2 ways Nyquist's *sampling limit* is different from Shannon's *sampling theorem*?

NOTES:

Periodicity

All the sampling theory discussed here applies to signals which are or will be considered periodic. "Will be considered" means that once a set of samples is taken, the mathematical analysis will consider that the signal was periodic, even if it was not. This places certain constraints on how the analyzed data is interpreted, especially on incomplete and non-coherent (discussed later) sample sets.

What does a sample set represent?

A sample set is a list of values that represent numbers. The origin of the samples can be a Waveform Digitizer which takes samples of a continuous analog signal. It can also be a set of samples that are mathematically created from an algorithm (normally via a computer) and sent to a Waveform Generator to create a signal. A sample set contains no information about time or frequency in the numbers. It is your job to understand the relationship of sample time and signal frequency and how they relate to the index of each sample value in a set, then properly process the sample values to extract information to compare to device test parameter limits.

The sample set taken with a WD can be as long or as short as necessary to gather enough numbers for adequate analysis. The sample set for a WG need only contain enough samples to accurately create the desired signal with a WG. In general, more samples are better from the perspective of digitizing or generating signals. In this chapter, the discussion of sampling applies to both digitizing and generating signals.

Question 6.2: Can the signal frequency be determined by knowing the following amplitude sample data?

0.7, 0.8, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4

NOTES:

Converting a time sample set to frequency

A benchtop instrument known as a *spectrum analyzer* displays the frequency characteristics of signals. The basic operation of a spectrum analyzer is like putting the signal into a bank of bandpass filters as shown in Figure 6.2.

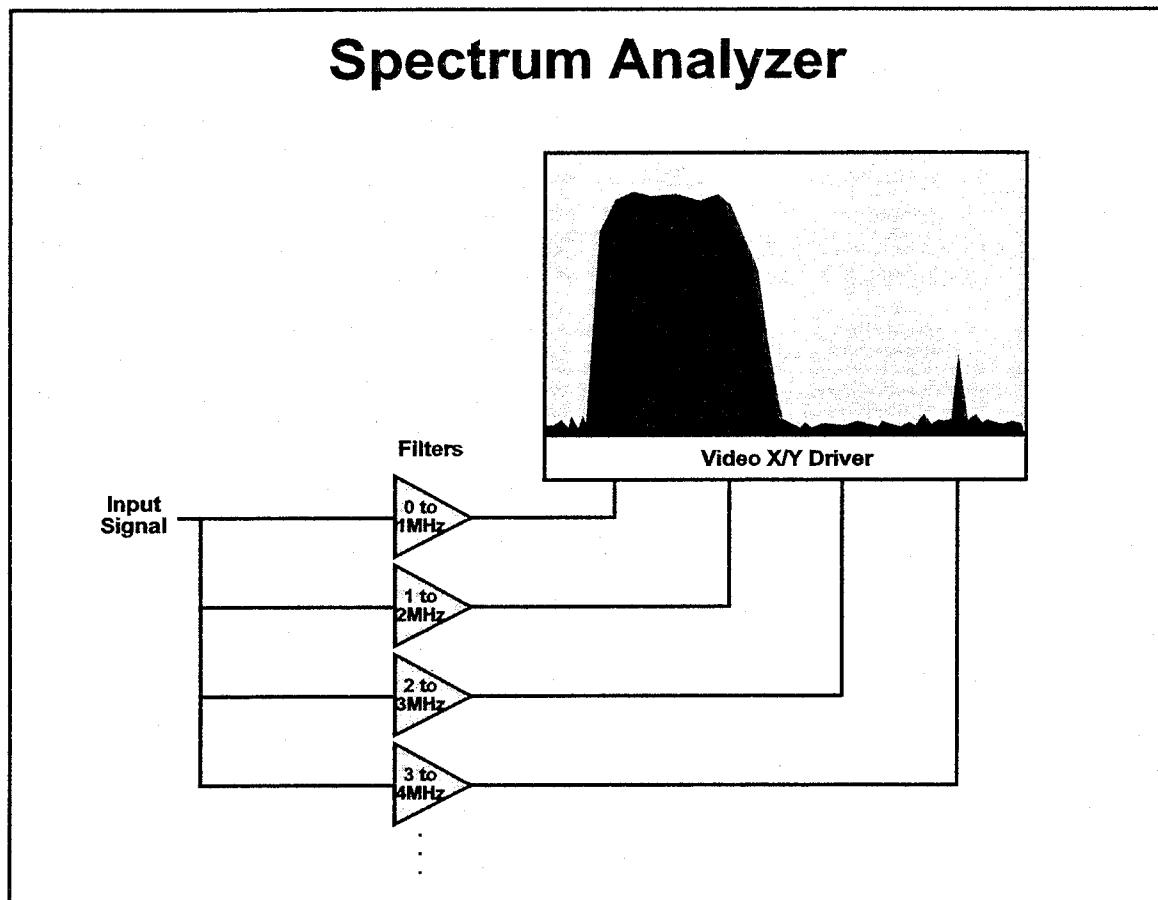


Figure 6.2

Each filter extracts the sinusoidal components in its bandpass range and sets the amplitude to be plotted for that band. The first benchtop spectrum analysis instruments worked this way, and some purely continuous

NOTES:

(non DSP type) spectrum analyzers still work this way. This type of spectral analysis is fast becoming obsolete, however, because DSP based spectral analysis offers many advantages over traditional techniques.

The section *Fourier series* on page 146 describes how a complex waveform can be analyzed as a sum of pure sinusoids and discusses a little of the mathematics required for the analysis. A set of discrete samples taken from a time waveform can also be analyzed and mathematically converted into a discrete set of frequency data which represents the sinusoidal frequency components of the waveform. The conversion of time to frequency data is performed with an algorithm known appropriately as the *Discrete Fourier Transform*.

Discrete Fourier transform (DFT)

The DFT is a mathematical algorithm which translates amplitude data taken in time into amplitude data versus frequency. It is the mathematical equivalent of a benchtop spectrum analyzer and takes amplitude versus time data and returns amplitude versus frequency data. Mathematically, the algorithm is a series summation of the product of each sample times a complex number. It is the discrete equivalent of the Fourier integral in equation (5.33) and is stated as

$$X(b) = \sum_{n=0}^{N-1} x(n)[\cos(2\pi nb/N) - j\sin(2\pi nb/N)] \quad (6.1)$$

where n = one of N samples, N = total number of samples, b = one of B frequency bins where each bin represents a frequency range of F_s/N and j = the phase (imaginary) operator. The DFT algorithm uses each sample point in the summation from 0 to $N - 1$ for each analyzed frequency. Note that all N sample points contain information about all B frequencies, thus each of the B frequencies for which information is desired requires a summation of N time sample products. Processing a DFT is slow, because N^2 calculations is necessary. For example, a 2000 point DFT requires 4 million calculations, often floating point calculations, which are slower than integer calculations.

NOTES:



Sampling

Fast Fourier transform (FFT)

The FFT remedies the DFT speed problem by skipping over portions of the summations which produce redundant information. A mathematical description of an FFT algorithm (there are many in common usage) is non-trivial and is not important to our use of it. Consider it as a tool you can use without knowing the details of how it works. Two facts are important to know for using the FFT:

1. The number of sample points must be a power of 2
2. The number of additions and multiplications is

$$\frac{N}{2} \log_2 N \quad (6.2)$$

which reduces the calculation count for a 2048 point FFT to 11000, a *lot* less than 4 million as required by a DFT.

Question 6.3: What is the difference between an FFT and a DFT?

Using an FFT algorithm

If you've never used an FFT, it seems to be a mysterious mathematical concept with no basis in real life. It is, however, a pragmatic tool which is used in mixed signal testing and in many other aspects of your life (e.g. virtually any telephone call). Using an FFT involves passing time data into the algorithm and receiving the returned frequency data.

NOTES:

The time data passed to an FFT algorithm consists of an array of real amplitude sample values taken from a signal (or created mathematically as in the DSP Lab software). The array is truly the array type you know from standard software coding. Because an FFT operates in complex number space, it returns frequency data as a real and imaginary frequency array. An example in C could look like:

```
void CreateMagArray
{
    double TimeArray[4096], RealFreq[4096], ImagFreq[4096], Mag-
Freq[4096];
    TakeSamples(TimeArray); /* ATE system routine to take 4K samples */
    PerformFFT(TimeArray, RealFreq, ImagFreq); /* Call FFT routine */
    GetMags(4096, RealFreq, ImagFreq, MagFreq); /* See below */
}
```

The PerformFFT routine passes a filled array of time samples and 2 empty arrays for frequency data. The routine fills the frequency arrays with real and imaginary values for each of N points; these represent rectangular coordinates for the complex frequency plane as seen in Figure 5.26 on page 156. To create a magnitude plot, each complex frequency pair must be converted to a magnitude value using equation (5.34) on page 158. This is done using the routine:

```
void GetMags(int Count, double RealFreq[], double ImagFreq[], double
Mags[])
{
    int Index;
    for (Index = 0; Index < Count; Index++)
    {
        Mags[Index] = Sqrt(RealFreq[Index] * RealFreq[Index] +
                           ImagFreq[Index] * ImagFreq[Index]);
    }
    return Mags;
}
```

Let the test system do the processing

Note that the DSP subsystem in your mixed signal tester will have optimized routines for much of this “number crunching.” For example, there is probably a built in routine which takes a set of time samples and returns a set of magnitude points. There is also probably a routine which takes a rectangular frequency array pair and returns a magnitude and phase array pair. *Know which built in routines are available and use them!*

NOTES:

Problems that can occur with sampling

Spectral replication and aliasing

Replication

By utilizing DSP techniques, a set of amplitude samples over time can be converted into a set of samples in the frequency domain. The frequency information returned from a DSP operation such as an FFT can be ambiguous regarding frequency. Mathematically, in the frequency domain, a sample set exists at the original test signal frequency (F_t) and is *replicated* at frequencies every k times the sample frequency, in both the positive and negative directions. Figure 6.3 illustrates this phenomenon; mathematically, the frequency components in the test signal are replicated to other frequencies as:

$$F_{replica} = F_t + kFs \quad (6.3)$$

where k is zero or any positive or negative integer.

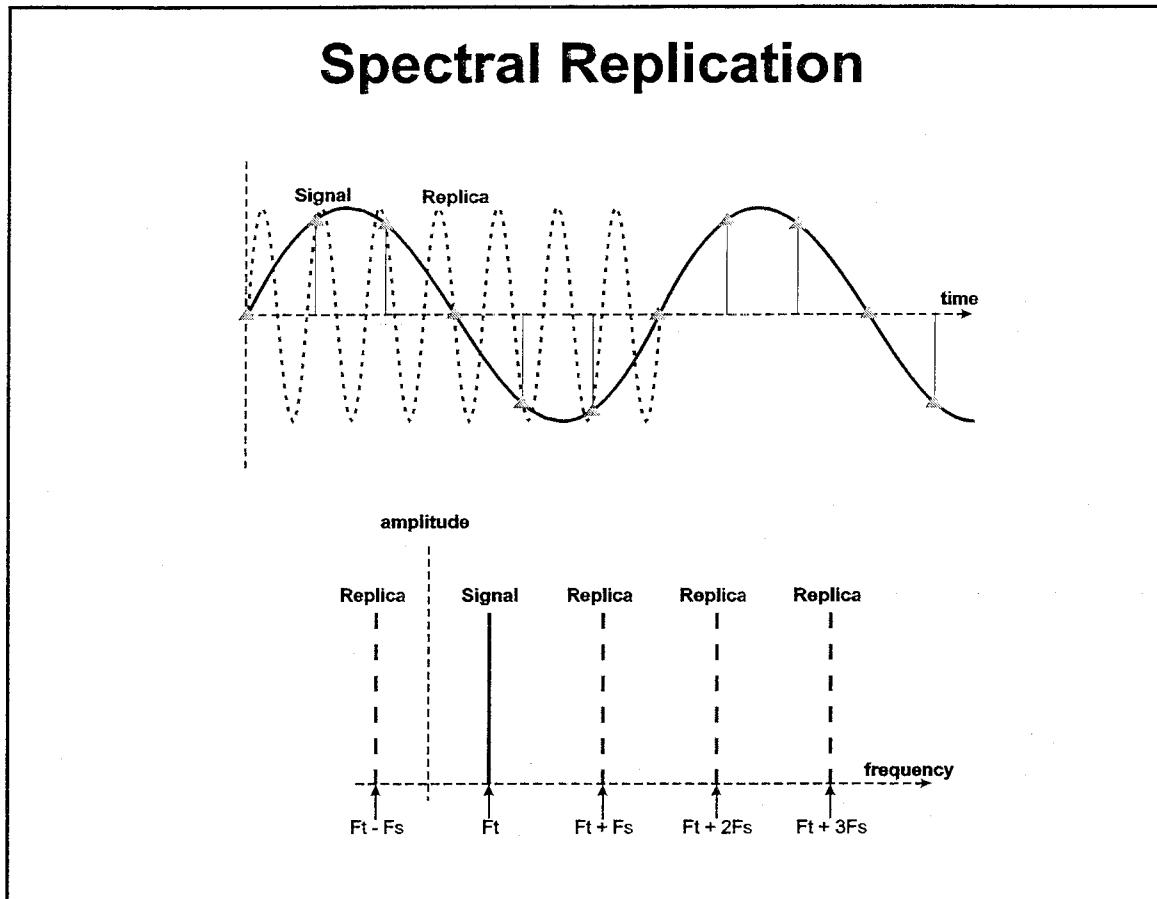
Notice how the sampled waveform has identical components at additional frequencies; these replicas extend to infinity. The information returned by a DFT/FFT can be analyzed using the "low pass" data, that is, the band of frequencies out to $F_s / 2$, ignoring the replicas.

Aliasing

Notice that the replicas extend to infinity in *both directions*, that is, a sampled signal with a component higher than those in the band of interest but lower than the sample frequency F_s can end up *aliased* into the frequency band of interest. This can be a problem—if the input signal being sampled contains signal components greater than $F_s / 2$ and less than F_s , those signal components are "folded" into the frequency data for the original signal, distorting the information returned by a DFT/FFT.

Although *replication* and *aliasing* can mean the same thing, we prefer to define "aliasing" as the distortion of frequency results due to signal components that are folded into the frequency band of interest from outside the Nyquist band.

NOTES:

**Figure 6.3**

An alias adds non-harmonically related frequency components to the sampled signal which cannot be distinguished nor removed from the signal of interest. In other words, a frequency component that is aliased into the frequency band of interest destroys the integrity of the information returned by the Fourier Transform. Notice in Figure 6.3 that, if the frequency band of interest goes from 0 to $F_s / 2$, the replica at $F_t - F_s$ is aliased into the low pass band of interest, corrupting the frequency data.

NOTES:

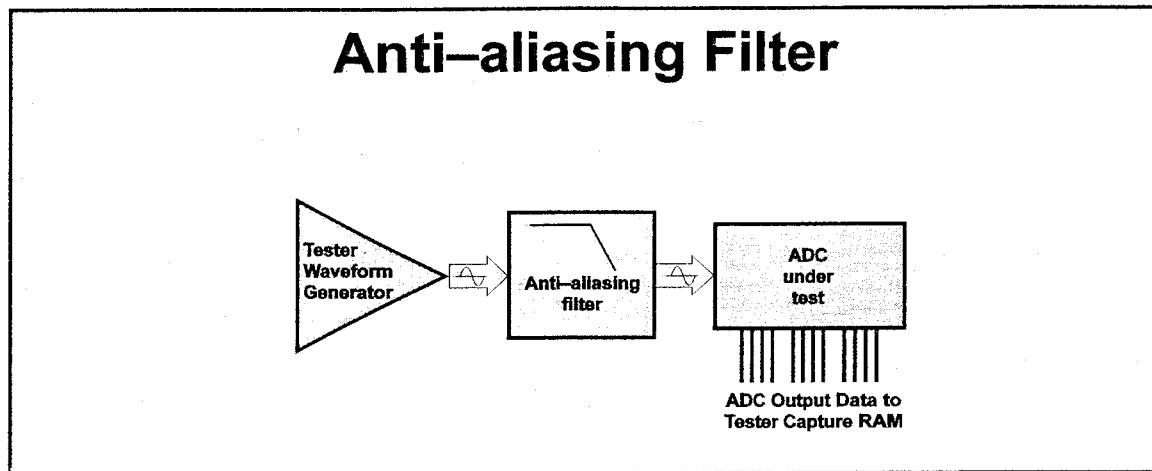


Figure 6.4

Prevention of aliasing errors

There is only one way to avoid the aliasing problem—remove frequency components greater than $F_s/2$ from the signal being digitized. This must be done with a continuous analog filter as shown in Figure 6.4. Any sort of switched capacitor filter will introduce its own high frequency noise and will not solve the problem. The distorted data exists in and corrupts the time samples so it cannot be removed with mathematical “digital filtering” via a DSP algorithm. To avoid it you *must* properly filter the input signal to the digitizer.

Question 6.4: What is the *only* way to prevent aliasing errors?

NOTES:

Leakage

Recall that Fourier transform algorithms assume that time data passed to them is periodic. Because the DFT / FFT returns a discrete set of amplitude points in the frequency domain, its results contain information only about frequencies that are integer multiples of the fundamental frequency F_s / N . The result of this and the assumption of periodicity is that frequency components contained in the sample set which are not integer multiples of F_s / N "leak" into the frequency points which are returned by the DFT / FFT.

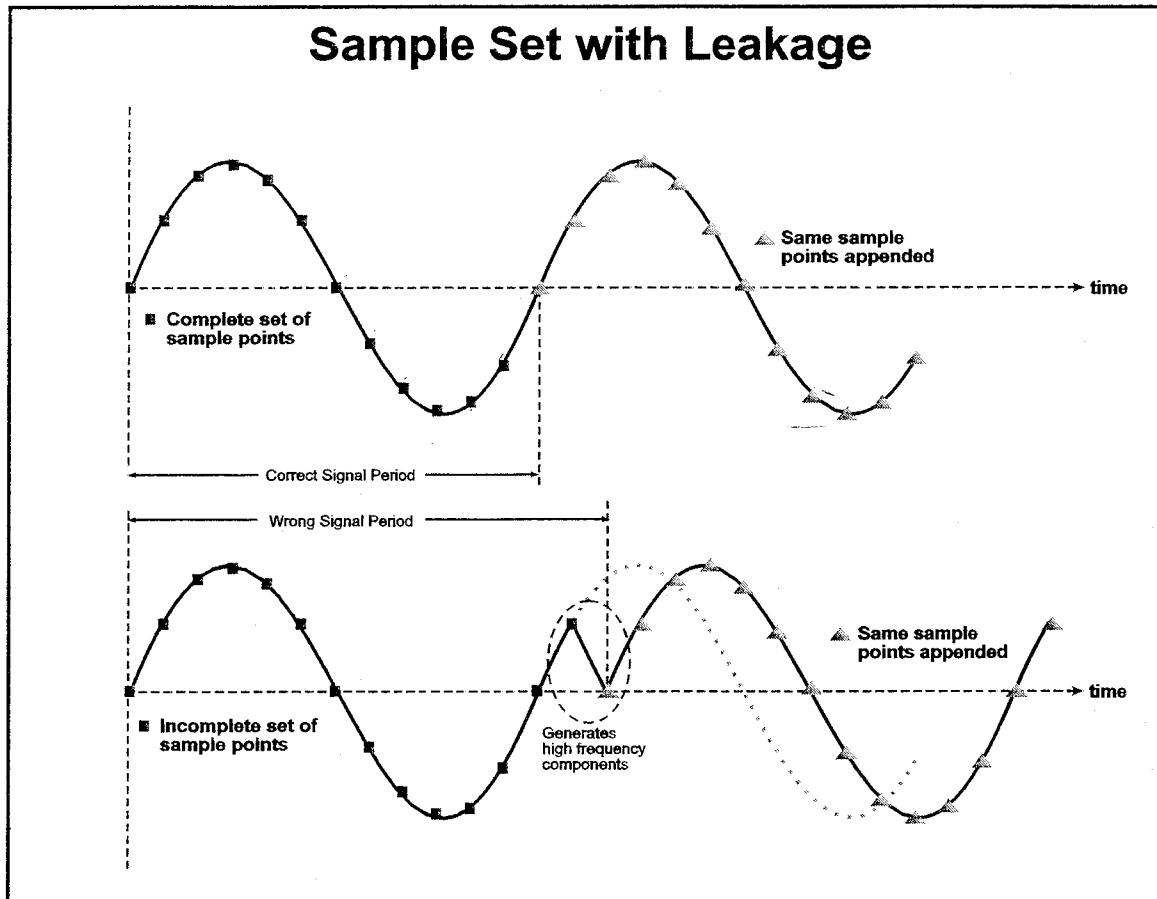


Figure 6.5

NOTES:

Software Testing Sampling

```
void HannWindow(int N, double TimeSamples[])
{
    int Index;
    double PI = 4 * arctan(1);
    for (Index = 0; Index < N; Index++)
        TimeSamples[Index] = TimeSamples[Index] * 0.5 *
            (1 - cos((2 * PI * Index) / (N - 1)));
}
```

The windowing functions available in a mixed signal ATE system will accept a time sample array and return a windowed sample set array automatically.

Question 6.5: What causes leakage?

Question 6.6: What can prevent leakage?

Question 6.7: What can be used to minimize leakage?

NOTES:

$\sin(x) / x$ Amplitude Error

Sampling theory presumes that a given sample in time represents a finite amplitude with zero time width. In general, the output from a DAC or WG remains constant between output value changes. This creates a waveform which can be modeled as a series of rectangular pulses of varying amplitude as shown in Figure 6.6.

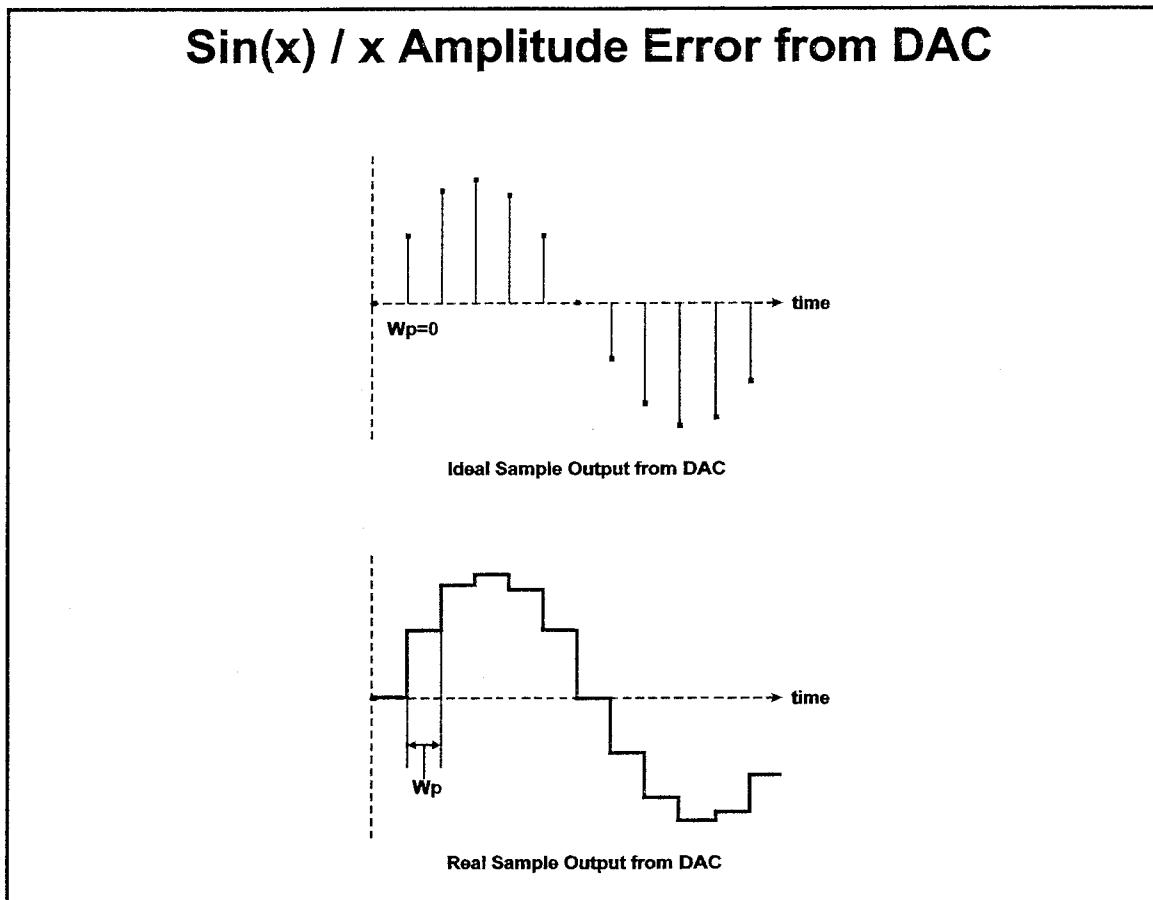


Figure 6.6

When the frequency conversion math is performed for this series of rectangles, the frequency spectrum of a sine wave is multiplied by the spectrum of a rectangle, which has a $\sin(x) / x$ characteristic. The magnitude spectrum of a rectangle is shown in Figure 6.7 in the top left diagram. The amplitude error versus frequency

NOTES:

for the generated sine wave can be seen in the other 2 diagrams. The error can be compensated with a filter or pre-compensated by modifying the data going to the waveform generator.

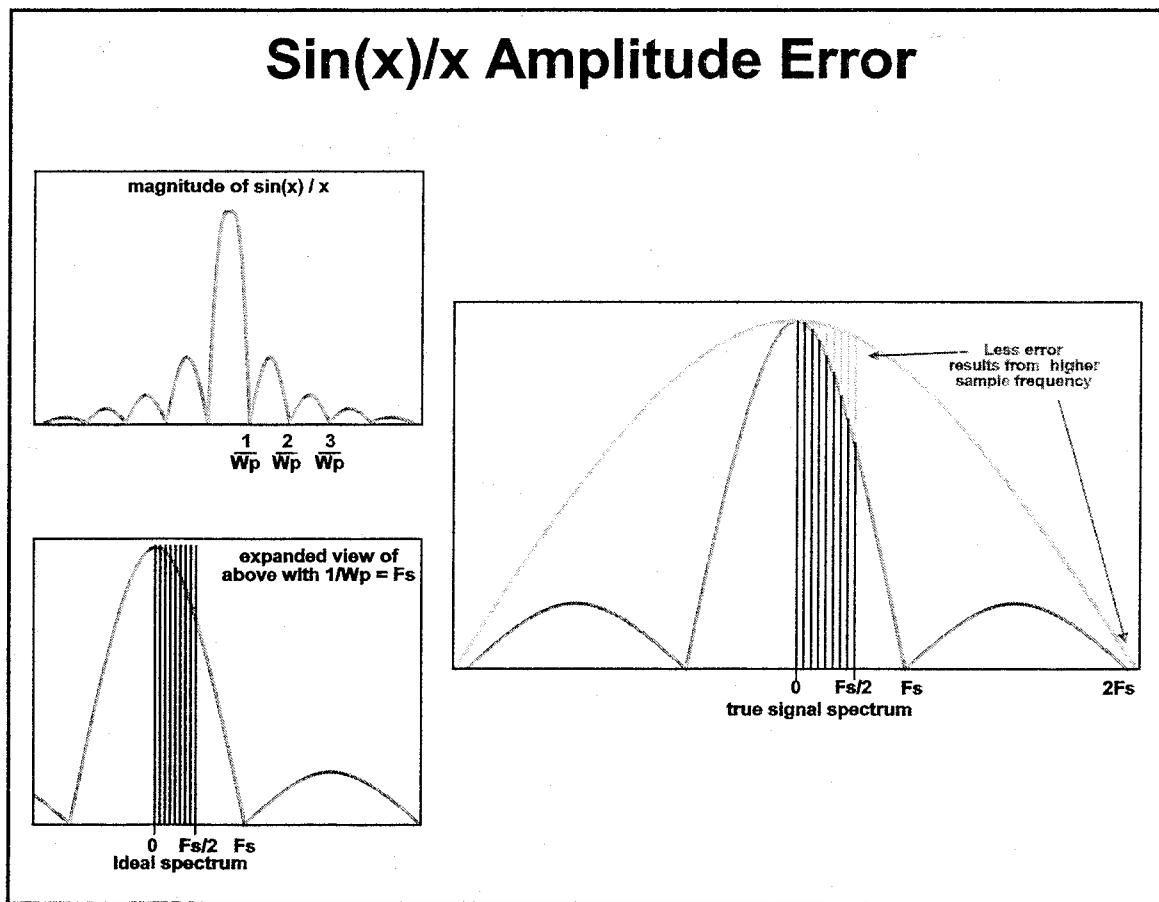


Figure 6.7

Notice that the pulse width in Figure 6.6 is the same as the sample period $1/F_s$, causing the frequency characteristic of the $\sin(x)/x$ curve to be zero at integer multiples of F_s . In a test situation, all our frequencies of interest will be between 0Hz (DC) and $F_s/2$. The error is an amplitude multiplication factor which is related to the ratio of signal and sample frequencies. The higher the ratio F_s / F_t the less amplitude error as seen in the large diagram in Figure 6.7. The multiplication factor is given by the $\sin(x) / x$ equation as related to F_s and F_t :

NOTES:



$$Multiplier_{Ampl} = \frac{\sin\left(\frac{\pi F_t}{F_s}\right)}{\frac{\pi F_t}{F_s}} \quad (6.4)$$

and can be given in dB or as a fraction. The amplitude multiplication factor for $F_s / F_t = 2$ (when the test signal contains a frequency component at the Nyquist frequency) is 0.6366 (-3.9dB) and 0.827 (-1.6dB) for $F_s / F_t = 3$. Note that the multiplier is 1 at DC and 0 when F_t is an integer multiple of F_s .

In Figure 6.7, it can be seen that for a given F_t , a higher F_s yields less amplitude error. The effect is more clear in the time domain—when the samples are closer together, the stepped DAC output more closely approximates a true sinusoid. As F_s goes to infinity, the frequency domain $\sin(x) / x$ curve in Figure 6.6 goes to horizontal and amplitude error goes to zero.

Question 6.8: How can $\sin(x) / x$ amplitude error be compensated?

Truncation error

Suppose you have a digitizer with infinite resolution; you still need to store and operate on the converted digital result. If you put it in a digital memory array with 16 bits, it allows integer storage of a quantity from 0 to 65535 (2^{16}). To calculate signal power, the voltage is squared, so each stored value gets squared. The squared result will need more bits than the original value, but when it is stored back into the 16 bit memory, all bits which extend past the end of the 16 bit register are truncated.

Although this is a simplified example of truncation error, it exists in any digitized signal. Digitized values are often represented using *floating point* values rather than fixed point (or integer) values. This reduces truncation errors but does not eliminate them. Truncation error virtually requires DSP processors to use double length registers in its multiplication, division and other operations.

Quantization Error

Another way analog information can be lost is in the digitizer. If an ADC is a 12 bit device then the best available digitizer resolution is 12 bits. Like any other physical system, this chain of data conversion is only as good as its weakest link.

NOTES:

The power of 2 to which an ADC can *resolve* a signal is the *resolution* of the digitizer, while the minimum quanta to which it resolves is the size of one *least significant bit*, called the *LSB size* or simply "one LSB." For example, an ADC which converts an analog signal to a 12 bit wide data word has "12 bits of resolution." Given a 12 bit ADC with a $\pm 5V$ full scale input range, what is its LSB size? The maximum number of steps between all outputs for a 12 bit device is $2^{12} - 1 = 4095$; the full scale range is $(5V - (-5V)) = 10V$; the LSB size is $10V / 4095 = 2.44mV$. Could you use this digitizer to directly measure the offset voltage of a unity gain connected OP-07 type op amp? Answer: no, because the OP-07 has input offset voltage in the $100\mu V$ range, too low for this digitizer to notice.

In this example, any analog signal change less than $2.44mV$ cannot be stored, or *quantized*, no matter how good the ADC. This minimum quanta associated with digitizing is known as *quantization error* and depends on the number of bits to which a signal is digitized and the digitizer's full scale range.

The maximum signal to noise ratio of an ADC depends directly on the quantization error as⁴

$$6.02 \times \text{bits} + 4.77 + 20\log\left(\frac{V_{in}}{V_{fs}}\right) \text{ dB} = \text{SNR} \quad (6.5)$$

where V_{in} and V_{fs} are the ADC input signal and full scale input range as RMS values. With a sine wave analog input signal whose peak value is equal to the ADC full scale input range (FSR), the maximum (ideal) SNR of the digitizer is:

$$6.02 \times \text{bits} + 1.76 \text{ dB} = \text{SNR} \quad (6.6)$$

Thus you can expect to measure an incoming analog signal's SNR to no better than this ideal value for the digitizer. This also means that, to see the best test results, you must condition your analog signal such that it matches the full scale range of the ADC in the digitizer.

Question 6.9: What is the best SNR (in dB) that can be achieved with a 13 bit ADC?

Slew Rate Error

Still another way a digitizer can lose information is if the analog signal is moving too fast for the digitizer. Given an analog signal that is a $\pm 1V$ peak sine wave, the maximum rate of change of the sine wave is when $t=0$ and the rate of change is given by its derivative. The derivative of $\sin(\omega t) = \omega \cos(\omega t)$; evaluated at $t=0$, $\cos(\omega t) = 1$ so the maximum rate of change is $1V * \omega$, where $\omega = 2\pi f$. With a signal frequency of 10KHz, the maximum

NOTES:

speed is approximately $62.8V/msec$. If the digitizer has 12 bit resolution and $5V$ full scale range, one LSB is $1.22mV$. To make a conversion of the sine wave at its fastest point, the conversion must be faster than $1.22mV / (62.8V \text{ per msec}) = 19\text{nsec}$. Yes, that says nanoseconds; we can definitively state that there are no known digitizers with a conversion speed in the tens of nanoseconds range.

Now you know why track and holds are used at the input of ADCs. They keep the input signal constant while a digitizer does its work. Now the requirement for capture speed has been moved away from the digitizer to a much faster device, the track and hold.

Jitter Error

As a signal changes in time, any deviation in time of the digitizer clock from its expected value (known as *jitter*) causes error. When a signal is digitized, only its amplitude information is preserved. Suppose you are digitizing a sine wave at its zero crossing and the sample frequency is the same as the sine wave frequency. In other words, you expect to get a value of zero for every sample. If the sine wave is perfect but the sample trigger signal jitters, the sampled values will be non-zero whenever the jitter causes the sample to be taken before or after the sine wave crosses zero.

When the time data is examined with a Fourier transform, the DFT/FFT algorithm assumes that the data was sampled at exact intervals, i.e. with no time jitter. Thus the data in the time sample array appears to the math algorithm as *amplitude* error, distorting the time waveform. Thus time jitter in a sampler (whether a generator or digitizer) appears as distortion in the frequency analysis results.

Suppose the sample timing is perfect (with no jitter) and the input sine wave has no amplitude distortion, but jitters in time...the result is the same. In other words, *jitter in either the sample timing or the analog signal timing can result in frequency distortion*. If there is sample and signal jitter that is statistically independent, the problem is compounded even further. If there is sample and signal jitter that is the same, they both move together so the problem is hidden; this is true when both the analog signal and the sample timing are created via a master clock.

Question 6.10: Is there any advantage to using one master clock in a test system? What is it?

NOTES:

Coherent sampling

What is coherent sampling?

Coherent sampling defines a way to guarantee that a sample set has a fixed, well defined relationship between the sample frequency F_s , the number of samples N , the test signal frequency F_t and the number of test signal periods sampled M . Coherent sampling restricts the timing relationships which exist between the test signal period and the sample period, making the timing setup for testing a particular device more difficult than with non-coherent sampling. It also requires a known waveform type, usually sinusoidal, as the input to a digitizer (ADC or WD) and the output of a generator (DAC or WG). If certain constraints are met, coherency also guarantees that the maximum amount of information about a particular waveform exists in the sample set (i.e. there are no duplicate samples on a waveform).

Why sample coherently?

Coherent sampling is a luxury afforded by those of us in the ATE industry because we have complete control over the analog signal to be generated or digitized. Those in the communications industry do not have the option of coherent sampling and must wrestle with problems involving leakage, non-periodic signals, windowing and more. Thanks to coherent sampling, mixed signal ATE can have faster test times, less computation and better results without any real drawbacks. The only complication is that timing synchronization of the test signal and sample clock frequencies is required, increasing test development effort.

When digitizing a waveform, coherent sampling eliminates the need for any time windowing by guaranteeing that the sample set contains a complete, periodic waveform representation. FFT output from a set of coherent samples puts the relevant information about the fundamental and harmonics into specific, well defined frequency ranges, called *bins*. When generating a waveform, there is no leakage because coherent sampling guarantees that there will be exactly a complete set of samples for one or more signal cycles.

Coherency relationships

F_s, N, F_t and M

The relationship which creates a coherent sample set is that the number of samples N and the number of test cycles M both be whole positive integer values and the sample parameters mentioned above conform to the relationship given by equation (6.7). In mixed signal testing, coherency means that a sample set contains an integer number of samples of an integer number of test signal cycles. No partial signal means no windowing is required. Coherent sampling also allows the most information to be collected about the test signal in the least amount of time. (This is also called "the fastest test time," a phrase you should know well.)

NOTES:

The equation for coherent sampling is one of the most important relationships in DSP based testing, and it is pretty simple:

$$\frac{Fs}{N} = \frac{Ft}{M} \quad \text{where} \quad (6.7)$$

- ◊ Fs = sample frequency
- ◊ Ft = test signal frequency
- ◊ N = total number of samples taken, **must be integer**
- ◊ M = total number of test signal cycles over which samples are taken, **must be integer**

Referring to equation (6.7), note that N samples may be taken from a waveform over M test signal periods. If $M = 1$, all samples are taken in a single test signal period. If $M > 1$, the samples are spread over more than one test signal period. The total time required to take all samples (the *unit test period*) requires M cycles of the test signal, which has frequency Ft .

UTP, Fourier frequency, frequency bins and resolution

A *unit test period* (UTP) is defined as the time required to take all samples and is given by

$$UTP = \frac{M}{Ft} = \frac{N}{Fs} \quad (6.8)$$

Referring to Figure 6.8, a DFT/FFT of a coherent sample set will produce $N + 1$ spectral components called *frequency bins*. The transform generates a mirror duplicate of the first $N/2$ points in the second $N/2$ bins, so the second half is discarded and only the first $N/2 + 1$ bins are used. The first bin (bin 0) is the DC component. The resolution of the spectrum (the width of each frequency bin) is determined by the UTP of the sample set and is called the *Fourier frequency* (FF) or *frequency resolution* ($Fres$):

$$FF = Fres = \frac{1}{UTP} = \frac{Fs}{N} = \frac{Ft}{M} \quad (6.9)$$

Since the total number of frequency bins is $N/2$ and the resolution of the frequency bins is the Fourier frequency, we can state that the maximum frequency in the spectral data is $Ft_{Max} = Fres \cdot (N/2)$. With $Fres = Ft/M$, we can see that the fundamental test frequency Ft will appear in the M^{th} frequency bin.

NOTES:

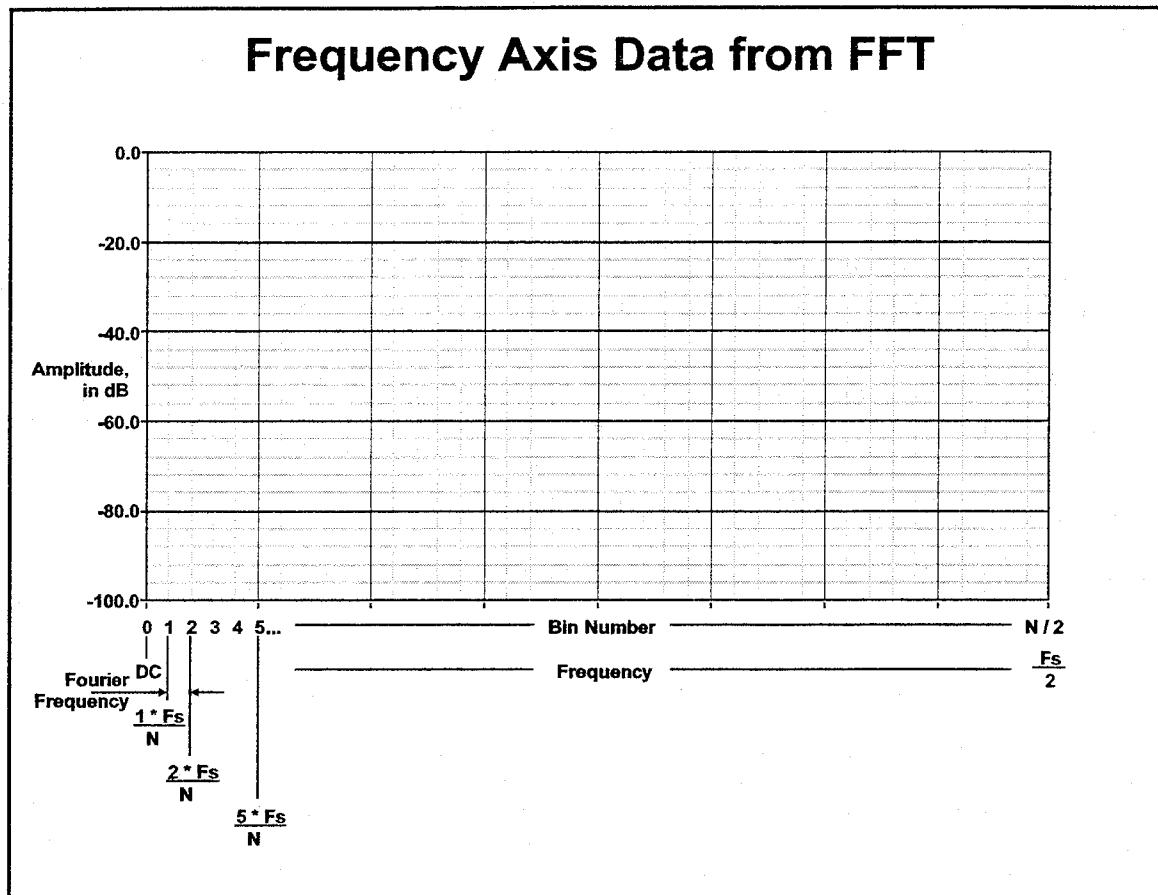


Figure 6.8

Question 6.11: If 128 samples are taken at 50 μ sec intervals from 7 complete signal cycles, a) what is the UTP? b) What is the Fourier Frequency? c) What is the signal frequency?

NOTES:

Mutually prime N and M assures unique sample points

When taking samples of a sine wave over more than one period, it is possible to sample at the same place on the wave only in a different signal period. When duplicate samples are taken this way, no harm is done to the sample data but no additional information is gained. In other words it wastes test time.

To avoid duplicate samples, make M and N *mutually prime*. Mutually prime numbers means that the 2 numbers have no common factors other than 1. For example, any 2 prime numbers are by definition mutually prime. Neither can be divided by a common value other than 1. Another example are 16 and 27; 16 is only divisible by 2 and 27 has 3 as its only factor; note that neither 16 nor 27 is a prime number.

When using the FFT, N , as a power of 2, is only divisible by 2. If M is chosen as an odd number, it is not divisible by 2. Thus *if you are using an FFT algorithm such that N must be a power of 2, choosing M as any odd integer will guarantee a coherent sample set with no duplicate points.*

Another benefit of coherent sampling

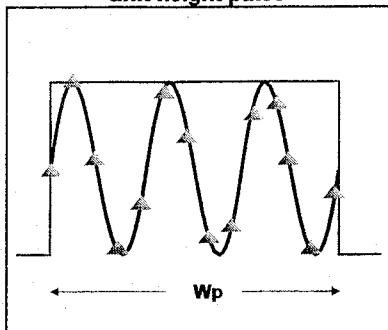
Now that we understand frequency bins, consider another aspect of sampling and how we benefit from doing it coherently. Recall from equation (5.33) on page 157 that the Fourier integral of a continuous function (or waveform) requires integration from minus to plus infinity. A sampled waveform contains only a small subset of the information contained in an infinite waveform because the samples are taken only from a small piece of the waveform. Figure 6.9 shows that a set of samples of a sine wave can be modeled as a continuous sine wave multiplied by a unit amplitude rectangular pulse the same length as the test signal cycle. Multiplication of two time functions is the same as convolution of their respective frequency functions.

The convolution operation modifies the single pure sine tone with the rectangle's $\sin(x)/x$ frequency characteristic. The width of the sample set and its implied rectangle is W_p . Notice that this is the same as the UTP and that frequency bins occur at integer multiples of F_{res} . The frequency characteristic of the rectangular pulse is known to be $\sin(1/W_p) / (1/W_p)$, which equals 0 at integer multiples of $1/W_p$, which is where all F_{res} frequency bins occur. With coherent sampling, problems with convolution due to a finite sized sample set are eliminated because the $\sin(x)/x$ curve is zero in all frequency bins of the FFT result.

NOTES:

Finite Sample Set Modeled as Signal x Rectangular Pulse

Sampled sine wave
inside rectangular
unit height pulse



Frequency domain
characteristic of pulse
($\sin(x) / x$)

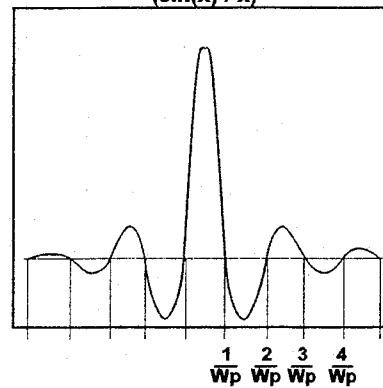


Figure 6.9

NOTES:

Digitizing samples

Conceptually, sampling is not a complicated process. You must decide 3 things—

1. How many samples of a waveform are required (N)
2. How often the samples should be taken ($1 / Fs$)
3. Over how many cycles of the test waveform they should be taken (M)

With these decisions made, put the signal into the digitizer, take the samples and store them. This section, rather than having a lot of words about sampling, concentrates on the act of sampling by doing it with the DSP Lab software.

Lab Exercise 6.1—Sampling a Waveform

Lab goals

- ❖ Create and view a continuous sine wave by making a Fourier series which has only 1 term
- ❖ Set sample parameters Fs , N , and M to create a specific Ft .
- ❖ View the samples of the continuous sine wave

Lab objectives

- ❖ Review Fourier waveform creation
- ❖ Show the relationship of sample equation parameters Ft , Fs , N and M .
- ❖ Illustrate how samples are taken from a continuous waveform
- ❖ Illustrate how changes to the sample parameters affect the test signal frequency

A note about the sampling parameters in the labs

The lab exercises involving sampling have entry boxes for Fs , N and M . Fs , the sample frequency, sets the rate of taking samples of the test signal. Fs is the inverse of the sample period and is sometimes called the *digital frequency*.

N is the number of samples taken; It is not possible to have a partial sample, so N is required to be an integer. The number of cycles M over which the samples are taken can be a real number with a fractional portion (e.g. $M = 3.25$). When M is an integer, sampling is coherent. When M is not an integer (in other words, the samples are taken over a unit time period (UTP) which is not a count of full test signal cycles), sampling is non-coherent and there is leakage in the DFT/FFT spectral result.

NOTES:

Software Testing Sampling

Initialize the lab and create a sine wave to sample

Begin the exercise by starting the DSP Lab application and pressing the *Sample Digitizer* button. Follow these steps:

1. Press the *Clear Harmonics* button to start fresh.
2. Set *Fundamental Frequency* to $1e3$ ($1 \times 10^3 = 1000\text{Hz}$). This may be changed later by sample parameter settings.
3. Set *Harmonic Number* to 1 and *Peak Amplitude* to 1.
4. Change to *Oscilloscope*.
5. Set *Amplitude* = 2V and *Time Scale* = 2msec.

You should see a single sine wave with a 1msec period. Note that the *Show Samples* checkbox is disabled.

6. Press the *Sampler* button; make sure it is set to *Calculate Ft*.
7. Set $F_s = 16e3$, $N = 16$ and $M = 1$. (Use the mouse or press Tab to change between entry boxes. Do not press the *OK* button).

Lab Question 6.1.1: What is the calculated value of F_t after the other parameters are set? (Press the Tab key if the value is not visible.)

Press *OK* to close the *Sampler*.

Lab Question 6.1.2: What is the period of the sine wave?

The *Show Samples* checkbox is now enabled. Click it with the mouse so it is checked. Red triangles show where samples are taken from the waveform.

Lab Question 6.1.3: How many samples are there? To which sample parameter does this correspond? (If necessary, reopen the *Sampler* to see.)

NOTES:

Open the *Sampler* again and set $M = 3$. Press the Tab key but don't close the *Sampler* yet.

Lab Question 6.1.4: What happened to Ft ? Why?

Close the *Sampler* with OK.

Lab Question 6.1.5: What is the frequency of the sine wave now?

Lab Question 6.1.6: How many samples are taken?

Lab Question 6.1.7: What is the total time required to take all samples?

Lab Question 6.1.8: How many signal cycles are taken over this period?

Lab Question 6.1.9: What is this period called?

NOTES:



Frequency analysis of samples

Once the samples are taken, the next step is to perform a time to frequency conversion on them and examine the frequency spectrum. Again we will use an exercise in the DSP Lab software to illustrate the principles discussed in earlier sections of this chapter.

Lab Exercise 6.2—Creating a Frequency Spectrum from Digitized Waveform Samples

Lab goals

- ❖ Create and view a continuous sine wave by making a Fourier series which has only 1 term
- ❖ Set sample parameters F_s , N , and M to create a specific F_t .
- ❖ View the samples of the continuous sine wave
- ❖ View a spectrum created by a Fourier transform of the samples
- ❖ Use both an FFT and a DFT to do the time to frequency conversion
- ❖ Modify the sample parameters to examine both a coherent and a non-coherent sample set
- ❖ Use windowing functions to improve the frequency spectrum of a non-coherent sample set

Lab objectives

- ❖ Review the process of digitizing samples of a continuous waveform
- ❖ Examine the spectrum created by a DFT/FFT algorithm from the digitized samples
- ❖ Illustrate the distinct time savings of using an FFT versus a DFT for spectral analysis
- ❖ Understand the difference between a coherent and a non-coherent sample set
- ❖ See the results of leakage on a frequency spectrum due to a non-coherent sample set
- ❖ See how windowing functions can reduce the effects of leakage

A note about the lab's use of DFT and FFT algorithms

If N is a power of 2, an FFT algorithm is used; if N is not a power of 2, a DFT algorithm is used. By setting a fairly high number of samples, e.g. 1024 then 1000, and comparing the calculation time for the FFT and DFT, you can see the time savings of the FFT over the DFT.

Initialize the lab, create a sine wave and sample it

Begin the exercise by starting the DSP Lab application and pressing the *Spectrum Analysis* button. Follow these steps:

1. Press the *Clear Harmonics* button to start fresh.

NOTES:

2. Set *Fundamental Frequency* to $1e3$ ($1 \times 10^3 = 1000\text{Hz}$). This may be changed later by sample parameter settings.
3. Set *Harmonic Number* to 1 and *Peak Amplitude* to 1.
4. Change to *Oscilloscope*.
5. Set the *Amplitude* knobs to 2V and *Time Scale* knobs to 1msec.

You should see a single sine wave with a 1msec period. Note that the *Show Samples* checkbox is disabled.

6. Press the *Sampler* button; set it to *Calculate Ft*.
7. Set $F_s = 5.12e6$, $N = 1024$ and $M = 1$. Press the Tab key to update *Ft* and notice that it has changed to 5KHz. Press OK to close the *Sampler*. Set *Show Samples* so it is checked.

Lab Question 6.2.1: The sampled section is a solid red sinusoid. Why?

Set the *Time Scale* to $500\mu\text{sec}$, then gradually decrease the *Full Scale* knob to $1\mu\text{sec}$ one increment at a time. Notice that there are individual samples being taken.

Lab Question 6.2.2: What is the (approximate) time between samples? To what parameter does this correspond in the *Sampler*?

Lab Question 6.2.3: What is the *Ft* frequency of the signal? (Press the *Sampler* button if you don't remember.)

Lab Question 6.2.4: Is this a coherent sample set? Are there any duplicate points? (Reopen the sampler if necessary to answer; you can press the *Cancel* button or the *Escape* key to close with no changes.)

NOTES:

Look at the frequency spectrum of the sample set

Change to the *Spectrum Analyzer*. Set the *Frequency* knob to 10 to show only the first 10 bins of the spectrum. Notice that when the mouse cursor is over the graph, the bin number, frequency and amplitude are displayed in the status bar at the bottom of the window. Remember that the values for frequency and amplitude are limited by the resolution of the screen pixels and are only approximate. The bin numbers are exact.

Lab Question 6.2.5: Place the cursor at the very top of the vertical frequency spike visible in the graph. What is the bin number? the frequency? the amplitude?

Lab Question 6.2.6: How does the frequency compare to F_t as noted in 6.2.3?

Look at the spectrum of a triangle wave, which has multiple sinusoidal components

Notice that the only frequency component visible in the graph is the single sinusoid as created by sampling the sine wave. Change to the *Fourier Waveform Generator* and press the *Triangle* button in the *Preset Waveforms* panel. This clears the sine wave that was created before and creates all harmonics out to the 20th for a triangle wave as given by equation (5.22) on page 147.

Change to the Oscilloscope, set *Time Scale* knobs = 1msec and observe the triangle wave with its components. Recall that equation (5.22) has only odd harmonics and that the *Fourier Waveform Generator* only has harmonics out to number 20.

Lab Question 6.2.7: How many odd harmonic terms will exist between harmonic 0 (DC) and harmonic 20 for this triangle wave?

NOTES:

Change the *Amplitude* knobs to 50mV and the *Time Scale* knobs to 50 μ sec. Set *Show Samples* so it is not checked.

Lab Question 6.2.8: Can you see all the harmonic terms? (Remember that there is also the sum of all harmonics plotted as the triangle wave itself, so the number of waves you see will be 1 more than you answered in Lab Question 6.2.7.)

Change to the *Spectrum Analyzer* and set *Frequency* to a maximum bin of 50.

Lab Question 6.2.9: Can you see all the harmonic terms as frequency spikes? How many? Which bins are they in?

Look at the time to calculate the frequency spectrum using DFT not FFT

Recall that the DSP Lab software uses an FFT algorithm when N is a power of 2. The previous sample set had $N = 1024$, which is a power of 2. Return to the *Oscilloscope*, press the *Sampler* button (set to *Calculate Ft*) and set $N = 1023$. Close the *Sampler* with the *OK* button or by pressing *Enter*. Change to the *Spectrum Analyzer*.

Lab Question 6.2.10: Did you notice how much longer the calculation took? Unless you are leaving for lunch, don't try this with 5000 samples!

Create a non-coherent sample set by setting M to a non-integer value

If M is the number of signal periods over which we are sampling, setting M to a non-integer value means we are sampling over part of a test signal period. Change to the *Oscilloscope* and set *Amplitude = 2V* and *Time Scale*

NOTES:

SoftICE Sampling Test

= 500 μ sec. Press the *Sampler* button, select *Calculate Ft*, set $M = 1.9$ and set N back to 1024 to save time. Set *Show Samples* so it is checked.

Notice the red sampled section of the triangle wave and how it no longer makes 2 complete cycles. Change to the *Spectrum Analyzer*.

Lab Question 6.2.11: Does the spectrum look any different? Remember how leakage spreads high frequency data from the incomplete sample period across all frequency bins? What is hiding the harmonics?

The more cycles of a signal that are sampled, the less effect the partial period will have compared to the total. In other words, sampling more periods decreases leakage. Change to the *Oscilloscope* and press the *Sampler* button; set $M = 11.9$ and press *OK*. Notice that the change of M changed Ft as we expect, but the UTP remains the same.

Return to the *Spectrum Analyzer* and set *Frequency* to a maximum bin = 500.

Lab Question 6.2.12: Can you see and count the harmonics now? How many?

Using a windowing function to improve spectral analysis

Set the *Window Function* drop down list to each windowing function one at a time. Change to the *Oscilloscope* and look at the samples for the different window functions. Notice how the time samples are changed and how the frequency spectrum changes for each different window. This illustrates that many window functions exist because some work better than others in different situations.

Lab Question 6.2.13: Which one do you think works best for this triangle wave?

NOTES:

Generating time samples

Often it is necessary to create data points for a waveform to be sent to a DUT. There are algorithms built into your mixed signal ATE system's WD for many of the common waveforms such as a sinusoid and possibly square, triangle and other waveforms. For those times when the required waveform is not available, you must know how to create your own. We will discuss 2 techniques for creating waveform points, using a software algorithm and using an Inverse FFT.

Using a software algorithm

A computer algorithm to calculate the points and store them in an array is a fairly straightforward approach to creating a waveform. For example,

```
void MakeSinePoints(double Freq, SamplePeriod)
{
    int N = 4096;
    double WavePts[4096];
    int n;
    PI = 4 * arctan(1);
    for (n = 0; n < N; n++)
        WavePts[n] = sin(2 * PI * Freq * (n * SamplePeriod)); // sin(2πft)
}
```

This is a good approach to generating waveform points when a single sinusoid or a tone that is the sum of 2 sinusoids is the desired waveform. The *sin* function ranges from -1 to +1 in amplitude, so it will probably be necessary to add some sort of scaling factor that each point is multiplied by; the factor depends on the required DUT input range and the WD output range.

This technique for generating a signal is also useful in testing a DAC and is discussed in detail in *Using a sine wave equation* on page 226.

Using an Inverse Fourier Transform

The mathematics of the DFT and FFT are reversible, that is, a set of frequency data can be passed to an Inverse FFT routine and processed, with a set of time samples returned. This can be useful in some test situations, especially when generation of a complex waveform is required.

NOTES:

When the IFFT is most useful

Suppose you need to test a filter circuit for bandwidth and are concerned primarily with test time and accuracy. The input signal can have several forms, each with its own requirements:

Input	Requirements	Problems	Benefits
Many individual sine waves	Sine waves must be sent to the DUT one at a time, each at a different frequency, to test the entire filter bandwidth.	Each sine wave must be generated, the filter must be allowed to settle and the output must be digitized. This means a s-l-o-w test.	Easy to program; outputs are easy to digitize.
A single swept wave	A frequency modulated waveform must be created that tests the entire filter bandwidth.	The FM wave may be difficult to create and coherently sample.	Fast test.
A single signal that contains many sine wave components	A tone waveform must be created that contains all the necessary frequency components to test the entire filter bandwidth.	Coherent sampling of output waveform may be complicated depending on how many sine terms are summed.	Fast test; easy to create filter input signal using Inverse FFT.

As noted in the last column of the last row, it is easy to create a tone containing multiple frequency components using an Inverse FFT. An array filled with frequency data, having very small values (e.g. -160dB) for the frequency points with no amplitude and large values (e.g. 0dB) for frequency points at the tone components.

How to use an Inverse FFT (IFFT) algorithm

The use of an IFFT algorithm is virtually identical to using a forward FFT. You fill an array with frequency data, pass it to the IFFT and an array of time samples is returned. Consider a filter under test that requires a bandwidth test for -3dB points at 1KHz, 2KHz and 4KHz. If we choose to use a 16 point IFFT which covers the frequency range of 0 to 10KHz, each frequency bin will cover $10K / 16 = 625\text{Hz}$. Table 6.1 shows a frequency array which could represent the data points for a 16 point IFFT for this filter. Recall that the Frequency Bin val-

NOTES:

ues represent the array subscript for the values in the frequency array. The Amplitude Value is what we put into the array to pass to an IFFT function.

Frequency Bin (Array Position)	Frequency (Hz)	Amplitude Value
0	DC	10^{-8}
1	625	1
2	1250	10^{-8}
3	1875	1
4	2500	10^{-8}
5	3125	10^{-8}
6	3750	1
7	4375	10^{-8}
8	5000	10^{-8}
9	5625	10^{-8}
10	6250	10^{-8}
11	6875	10^{-8}
12	7500	10^{-8}
13	8125	10^{-8}
14	8750	10^{-8}
15	9375	10^{-8}
16	10000	10^{-8}

Table 6.1

Notice that the fundamental, which exists at bin 1 as usual, is given an *Amplitude Value* = 1. This will be the 0dB level.

Question 6.12: The bins with *Amplitude Value* = 1e-8 are how many dB below 1V?

NOTES:

Limitations of using the Inverse Fourier Transform

Both the IFFI and the IDFT have limitations. One is that the amplitude of the returned data must be scaled; it does not equal anything in particular and the peak value depends on the specific algorithm used (some DFT/FFT algorithms use a different overall divisor than others).

Another limitation is related to the frequency points sent to the IDFT/IFFT. The only frequency points that can be represented are those of a specific frequency bin. By using the inverse transform when it makes the most sense, this limitation is not normally a problem.

Finally, recall that the Fourier transform assumes that the time data is periodic. Thus when passing in a set of frequency points, an inverse transform will return points for a complete waveform period; no partial period data can be created with an IDFT or IFFT.

Lab Exercise 6.3—Generating Time Samples with Inverse Fourier Transform

Lab goals

- ❖ Create and view a continuous sine wave by creating a frequency spectrum and doing an inverse Fourier transform

Lab objectives

- ❖ Learn about the inverse Fourier transform
- ❖ Create a waveform with inverse Fourier transform
- ❖ Examine the time characteristics of the created waveform

Using the Inverse FFT screen

This lab allows you to put frequency data into a spectrum graph with the mouse. The points you place in the graph become points in an array of frequency data that will be passed into an inverse Fourier transform algorithm. All points that you do not set in the graph are initialized to -160dB so they are not visible on the graph and are essentially zero. The actual values are Vmax = 1 and zeroed values are set = 1e-8; using 1e-8 allows dB calculations without concern of getting a math error because the logarithm of 0 is undefined. (See page 127 of the *Properties of logarithms* section.)

To set a point, notice that the bin, frequency and amplitude data are visible in the status bar when the mouse moves over the graph. Find the point where you wish to place a frequency component then click once with the mouse. If you click at the wrong place, go to the bottom of the graph directly under the erroneous point and click to set the point back to 0.

NOTES:

When time domain amplitude data is created with an IFFT, the x axis time values are an index of the data point in the array. The frequency of the output wave is determined by the update rate of the data according to the usual equation

$$\frac{Fs}{N} = \frac{Ft}{M} \quad (6.10)$$

Ft is the test frequency which is desired for the generated signal, N is the number of samples in the time domain amplitude data created by the IFFT operation, M is the number of test wave cycles in the waveform and Fs is the sample frequency at which the generated wave output is updated.

Note that this lab passes only magnitude points to the IFFT routine, which presumes a 0° phase shift for all points. Thus the returned time data contains a set of cosine and a set of sine points, respectively, in the real and imaginary arrays.

In the DSP Lab software, press the *WG from Inverse FFT* button.

1. Set the *Frequency* knob to a maximum bin of 10 to get the best accuracy for the mouse cursor.
2. Open the *Sampler* and be sure it is set to *Calculate Ft*.
3. Set $Fs = 5.12e6$, $N = 1024$ and $M = 1$.
4. Close the *Sampler* with the *OK* button.
5. Move the mouse cursor to *Bin = 1* and *Amplitude = 0dB* by watching the values in the bottom status screen.
6. Click once to set a frequency point. (Note: because of the graph's resolution, the frequency point looks more like a triangle than a spike, but there really is only one data point.)

Lab Question 6.3.1: What is the frequency of bin 1?

Change to the *Oscilloscope* tab and set *Amplitude* to 200mV and *Time Scale* to 500 μ sec.

NOTES:

Sampling

Lab Question 6.3.2: What do you see? What is the period? What is the amplitude? (Remember that amplitude is a limitation using IFFT and may require adjustment before using the generated points.)

The time waveform is created by points returned from the IFFT algorithm. To see the points, press the IFFT Points button. You can resize the window to see more points at one time, and use the scroll bar to move the list up or down.

Lab Question 6.3.3: How many points are in the list? This corresponds to which sample parameter?

Experiment with random or selected points on the Inverse FFT graph to see what time waveform results from them. Change F_s , N and M and set frequency points then check the results in the Oscilloscope window.

The magnitude values of the points generated with an IFFT function will depend on the specific IFFT algorithm. It may be possible to specify a peak value, a starting point, a phase, etc. or the data may come back as raw rectangular complex arrays (e.g. $a + jb$) that you must scale, convert to binary for DAC input, etc. For a pure single frequency sine wave, the IFFT does not have much benefit over writing an algorithm to generate DAC input points, but if you must create a complex wave containing several frequency components (e.g. for an IM test or a band of frequencies to test a filter), an IFFT can make the task easier.

NOTES:

Key points of this chapter

- ❖ You must sample at a frequency higher than twice the maximum frequency of interest in a signal
- ❖ Use an FFT to process samples if possible; it is much faster than a DFT
- ❖ Using an FFT requires that the number of samples be a power of 2
- ❖ $F_s / N = F_t / M$ where F_s = sample frequency, F_t = test signal frequency, N = number of samples taken, M = number of test signal cycles over which N samples are taken
- ❖ Coherent sampling requires N and M be integers
- ❖ Coherent sampling with mutually prime N and M gives the fastest possible test time with the most amount of information per sample set
- ❖ Leakage is caused by a sample set with an incomplete period
- ❖ Leakage can be reduced with a windowing function
- ❖ Leakage does not occur in a coherent sample set
- ❖ When generating signals with a DAC, $\sin(x)/x$ amplitude error occurs. It can be corrected with a filter or by pre-compensating the digital input data to the DAC.
- ❖ An inverse Fourier transform can be used to generate a set of time samples from a set of frequency data
- ❖ Using IFFT to generate time samples is best when multi-tone signals are required

NOTES:

Answers to chapter questions

Answer 6.1: 1) Shannon's theorem applies to infinite length continuous functions and Nyquist's to finite length portions of a function and 2) Shannon's theorem states that *only 2 samples per maximum frequency component* are enough and Nyquist's limit states that *more than 2 samples per maximum frequency of interest* are required.

Answer 6.2: There is no frequency associated with samples unless you know the sample frequency and number of cycles in the waveform.

Answer 6.3: The FFT is a "sports car" version of the DFT algorithm that is very fast by eliminating redundant calculations; it requires that the number of samples be a power of 2. The DFT is a "family car" that is much slower than the FFT but can be used on a sample set with any number of samples.

Answer 6.4: The use of a continuous time analog filter to remove frequency components $> F_s / 2$ from the signal being digitized.

Answer 6.5: An incomplete sample set, i.e. a set of sample points which do not form one complete cycle of a signal.

Answer 6.6: Creating a complete sample set by using coherent sampling.

Answer 6.7: Using Windowing functions on the time data *prior to* performing a Fourier transform.

Answer 6.8: With a filter.

Answer 6.9: 80dB

Answer 6.10: Yes. Any jitter in that clock becomes a "common mode" error and is not visible in sample data.

Answer 6.11: a) 6.4msec b) 156.25Hz c) 1093.75Hz

Answer 6.12: -160dB (160dB below 1V)

NOTES:

Answers to Lab Exercise 6.1

Answer 6.1.1: 1000Hz

Answer 6.1.2: 1msec

Answer 6.1.3: 16 samples, corresponds to N

Answer 6.1.4: Ft changed to 3000 as it must when M gets multiplied by 3.

Answer 6.1.5: 3000Hz, the same as Ft

Answer 6.1.6: Still 16 samples; N has not changed.

Answer 6.1.7: Samples are taken over a 1msec period

Answer 6.1.8: Samples are taken from 3 signal periods ($M = 3$)

Answer 6.1.9: This is a UTP of 1msec.

NOTES:

Answers to Lab Exercise 6.2

Answer 6.2.1: There are so many samples that the red triangles overlap on the graph

Answer 6.2.2: $0.195\mu\text{sec}$ is the time between samples (the sample period). It corresponds to $1/F_s$

Answer 6.2.3: 5KHz

Answer 6.2.4: Yes, it is a coherent sample set because M and N are both integers. There are no duplicate points in the set because M and N are mutually prime, i.e. they have no common factors.

Answer 6.2.5: Bin = 1, Frequency = 5000, Amplitude = 0dB

Answer 6.2.6: It is the same

Answer 6.2.7: 10

Answer 6.2.8: Yes, I see 11 waves. The black one going up is the triangle wave that is the sum of all the other waves.

Answer 6.2.9: Yes, 10 frequency components. They are in the odd frequency bins 1, 3, 5...19

Answer 6.2.10: Yes, calculation took much longer

Answer 6.2.11: The spectrum looks very different. The harmonics are no longer visible because the discontinuity in the sampled data for the partial period causes the high frequency information to be spread across all spectral bins. In other words, *leakage*!

Answer 6.2.12: Yes, there are 10 harmonics, but the leakage keeps the "floor" of the spectral data near -60dB. (And keep in mind that this is perfect, noise free sample data.)

Answer 6.2.13: The *Blackman-Harris* window gets my vote, although the *Hann* or *Blackman* run a close second. It depends on what you are trying to measure—SNR requires something with the lowest noise floor, whereas THD may need the most narrow fundamental and harmonic characteristic.

NOTES:

Answers to Lab Exercise 6.3

Answer 6.3.1: 5000Hz

Answer 6.3.2: A sine wave. The period = 200 μ sec, the amplitude = 0.117V

Answer 6.3.3: 1024 points (0 to 1023 is a count of 1024). This corresponds to $N = 1024$.

NOTES:

References

1. C. E. Shannon, "Communication in the Presence of Noise", *Proceedings of the Institute of Radio Engineers*, Vol. 37, 1949, pg 10.
2. Matthew Mahoney, *DSP Based Testing of Analog and Mixed Signal Circuits*, IEEE Computer Society Press, 1987, pg 37.
3. *Understanding Digital Signal Processing*, Richard G. Lyons, Addison Wesley Longman, Inc., 1997, pg 86.
4. *Ibid*, pg 362.

NOTES:

Digital to Analog Converter Dynamic Parameters

Goals

- ❖ Examine DAC dynamic specifications
- ❖ Explain how DSP spectral results are used to calculate dynamic parameters—SNR, THD, SINAD, IM
- ❖ Learn how to generate a sine wave with a DAC
- ❖ Understand why conditioning and filtering of DAC output is necessary
- ❖ Know why test system digitizer specifications are important
- ❖ Learn how to use a test system digitizer to capture a conditioned sine wave from a DAC
- ❖ Understand synchronization issues with regard to coherent testing

Objectives

- ❖ Have the student understand the dynamic parameters contained in a DAC specification
- ❖ Have the student recognize the complex relationship between the ATE system and the DUT, specifically in relation to an analog signal and a waveform digitizer
- ❖ Enable the student to solve real world problems related to digitizing any analog signal (not just DAC output)
- ❖ Enable the student to use tester or external conditioning circuitry to properly condition analog signals before digitizing
- ❖ Use questions and interactive software laboratory exercises to demonstrate the often difficult concepts of coherent sample generation, Fourier analysis and dynamic parameter calculation



What you will learn

- ◊ What dynamic DAC specifications represent with respect to what can be measured on a test system
- ◊ Test system configuration for DAC dynamic parameter testing.
- ◊ How to create the required DAC output signal for dynamic tests.
- ◊ How to properly condition the DAC output signal before digitizing it.
- ◊ Reasons why the DAC output must be conditioned.
- ◊ How to capture DAC output with a WD.
- ◊ How to process digitized DAC output samples to perform frequency analysis.
- ◊ How to use the frequency data to calculate dynamic parameters SINAD, THD and SNR.
- ◊ How to create a coherent multi-sinusoidal tone for use in filter testing, IM testing, et al.
- ◊ The sort of synchronization issues that can arise with simultaneous generation and digitizing of signals that must be coherent.

NOTES:

DAC Dynamic Specifications

A digital to analog converter, or "DAC," is a circuit that converts a digital number into an analog voltage or current level. A DAC can have one of many different architectures: resistor ladder, capacitor ladder, fully decoded, et al. It can have an internal reference, require an external reference, have voltage out or current out, unipolar or bipolar output range, serial data in, parallel data in, binary or BCD input format, latched inputs, and other variations. In short, it is a complex device.

Figure 7.1 illustrates some of the variations in pinout that may be found in a DAC data sheet. Notice that if the I_{out} pin is shorted to the *Summing Junction* pin, this DAC changes from a current out to voltage out device. There are other possibilities such as pins that allow bipolar operation (the output signal goes positive and negative with respect to common (or ground). Weighted capacitor DACs may require a clock input. In addition to the digital data input pins, other digital inputs may include a clock for serial data input or byte-by-byte data input and microprocessor interface signals such as chip enable or write. As with any circuit, you must carefully study the data sheet to know how to connect the DAC under test and provide the stimulus necessary to get the required signals from it for testing.

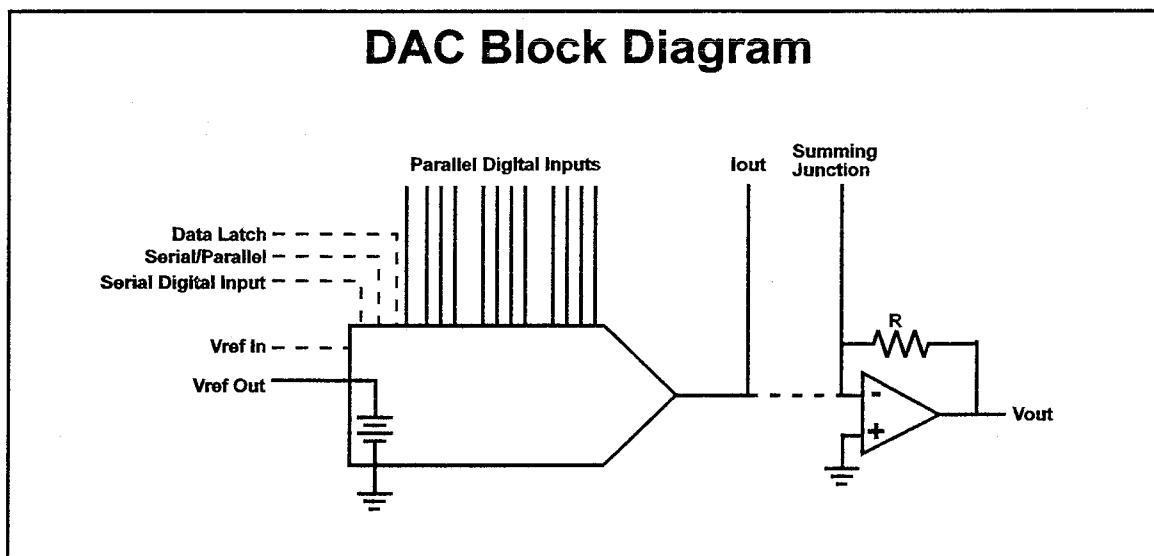


Figure 7.1

NOTES:



Digital to Analog Converter Dynamic Parameters

DAC dynamic specifications are:

- ❖ Signal to noise ratio (SNR)
- ❖ Signal to noise and distortion ratio (SINR or SINAD)
- ❖ Total harmonic distortion (THD)
- ❖ Intermodulation distortion (IM)

Also of interest are two specifications related to the DAC's maximum performance

- ❖ Maximum conversion rate
- ❖ Settling time

NOTES:

Spectral Components

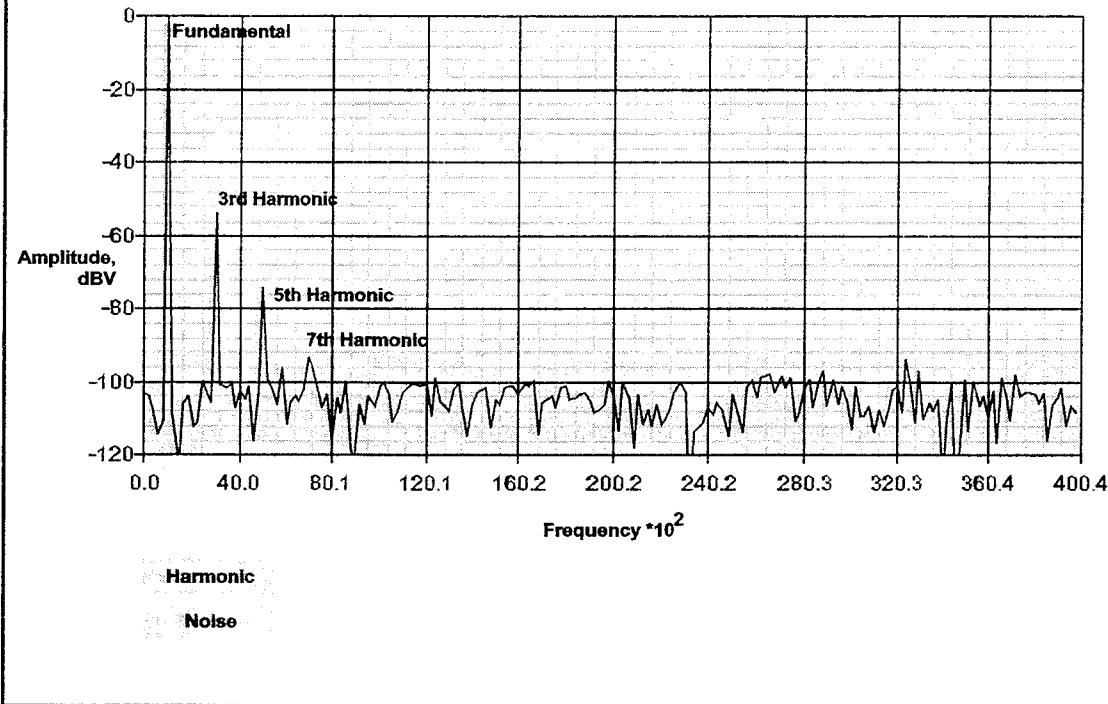


Figure 7.2

NOTES:

Signal to noise and distortion ratio (SNDR or SINAD)

SNDR is the ratio of the fundamental to the combined noise and distortion. Specified in dB. (See SNR and THD discussion for more information.)

Calculating Signal to Noise and Distortion (SINAD or SNDR)

This represents a ratio of the voltage at the fundamental frequency to the sum of the voltage components at all other frequencies. We start with the real and imaginary data returned from the FFT calculation and take the RMS sum of the signals in all bins except the fundamental (bin M). Then divide that root sum into the fundamental value as calculated earlier (Gain - Offset) and convert to dB. As an equation:

$$SINAD = 20 \log \left(\frac{amplitude_{BinM}}{\sqrt{\sum_{b=1}^{N/2} (amplitude_{(Bin=b, b \neq M)})^2}} \right) \quad (7.11)$$

Note that the denominator summation omits the bin 0 term, which is DC and is not considered noise nor is it harmonic data. Also, note that the summation omits the amplitude at bin M, which is the fundamental. Bin M is the amplitude term in the numerator. Perhaps more easily understood as a software algorithm, this equation is:

```
double SumSquared = 0;
int Bin;
for (Bin = 1; Bin <= N / 2; Bin++) /* Calculate denominator */
{
    if Bin != M /* Do not sum the fundamental */
        SumSquared += Amplitude[Bin] * Amplitude[Bin];
}
SINAD = 20 * log10(Amplitude[M] / sqrt(SumSquared)); /* dB of ratio */
```

This calculation is the RMS of all the noise and harmonic amplitudes *except the fundamental* divided into the fundamental then converted to dB. Often your mixed signal test system will have a built-in routine to calculate SINAD given arrays of real and imaginary frequency values. Use it.

NOTES:

Signal to noise ratio (SNR)

SNR is conceptually the same parameter as that for an op amp, and, like THD, is analyzed by presenting a digital version of a full scale sine wave to the DAC. The DAC output is filtered such that the fundamental is smoothed then removed, along with all harmonic components. Any remaining signal is considered noise and SNR is the ratio of the full scale fundamental to the summation of all noise. If harmonic signal components are not removed, this specification is SINAD, or signal to noise and distortion ratio. Specified in dB.

Calculating Signal to Noise Ratio (SNR)

Signal to noise ratio is a subset of SINAD, in which the components for harmonic distortion are not included in the calculation. Its calculation is a combination of the SINAD and THD calculations that includes the noise components but excludes the harmonic components at bin = kM :

$$SNR = 20 \log \frac{amplitude(rms)_{binM}}{\sqrt{\sum_{Bin=1}^{N/2} (amplitude_{Bin})^2, Bin \neq kM, k= 1, 2, 3...}} \quad (7.12)$$

Note that this calculation has a small error because the noise at the harmonic bins is excluded. As a software algorithm, Figure 7.12 is:

```
double NoiseSumSquared = 0;
int M = FtCycles;
int Bin;
for (Bin = 1; Bin <= N / 2; Bin++) /* Calculate noise */
{
    if (Bin mod M) != 0 /* Skip fundamental and harmonics bins */
        NoiseSumSquared += Amplitude[Bin] * Amplitude[Bin];
}
SNR = 20 * log10(Amplitude[M] / NoiseSumSquared);
```

NOTES:

Total harmonic distortion (THD)

While conceptually the same parameter as THD for an op amp, measurement of THD is different for a DAC because of the discrete nature of its analog output signal. The digital codes which represent points of a sine wave are presented to the DAC input; its output is a stepped equivalent of the sine wave and must be smoothed with a filter. The smoothed output is then analyzed in the frequency domain to find any signal components which are harmonically related to (are an integer multiple of) the fundamental frequency of the output sine wave. Specified in dB.

Calculating Total Harmonic Distortion (THD)

Total harmonic distortion is a ratio similar to SINAD, with 2 major differences:

1. THD does not include noise bins, only harmonic bins.
2. THD is a ratio in which the fundamental is in the denominator, not the numerator.

Mathematically, THD is expressed as

$$THD = 20 \log \left(\frac{\sqrt{\sum_{\substack{Bin = kM \\ k=2}}^{N/2} (amplitude_{Bin})^2}, k= 2, 3, 4...}}{amplitude(rms)_{binM}} \right) \quad (7.13)$$

Notice that the voltage level of the fundamental is in the denominator, so THD is a negative quantity (unless something is terribly wrong and the distortion is greater than the fundamental signal). The amplitude sequence starts at the 2nd harmonic (bin = $2M$) and sums the squared harmonic values. The sum is divided by the value of the fundamental and converted to dB. This equation ignores the DC term at bin zero and assumes the maximum frequency of interest is the Nyquist frequency ($F_s/2$). As a software algorithm, THD is calculated as:

```
double SumSquared = 0;
int M = FtCycles;
int Bin;
for (Bin = 2; Bin <= N / 2; Bin++) /* Sum squared harmonic terms */
{
    if (Bin mod M) == 0 /* Sum only harmonics bins */
        SumSquared += Amplitude[Bin] * Amplitude[Bin];
}
THD = 20 * log10(sqrt(SumSquared) / Amplitude[M]);
```

NOTES:



Intermodulation Distortion (IM)

IM is a test for non-harmonic product terms that appear in a device signal due to undesired modulation of two frequency components of a signal. This modulation is a result of non-linear characteristics within the device under test. The test is performed by putting a summed two sinusoid tone into a device and looking for frequency components in the sum and difference frequency bins of the two sine frequencies. Second IM product terms are found at $(F_1 \pm F_2)$; third harmonic product terms at $(2F_1 \pm F_2)$, etc.

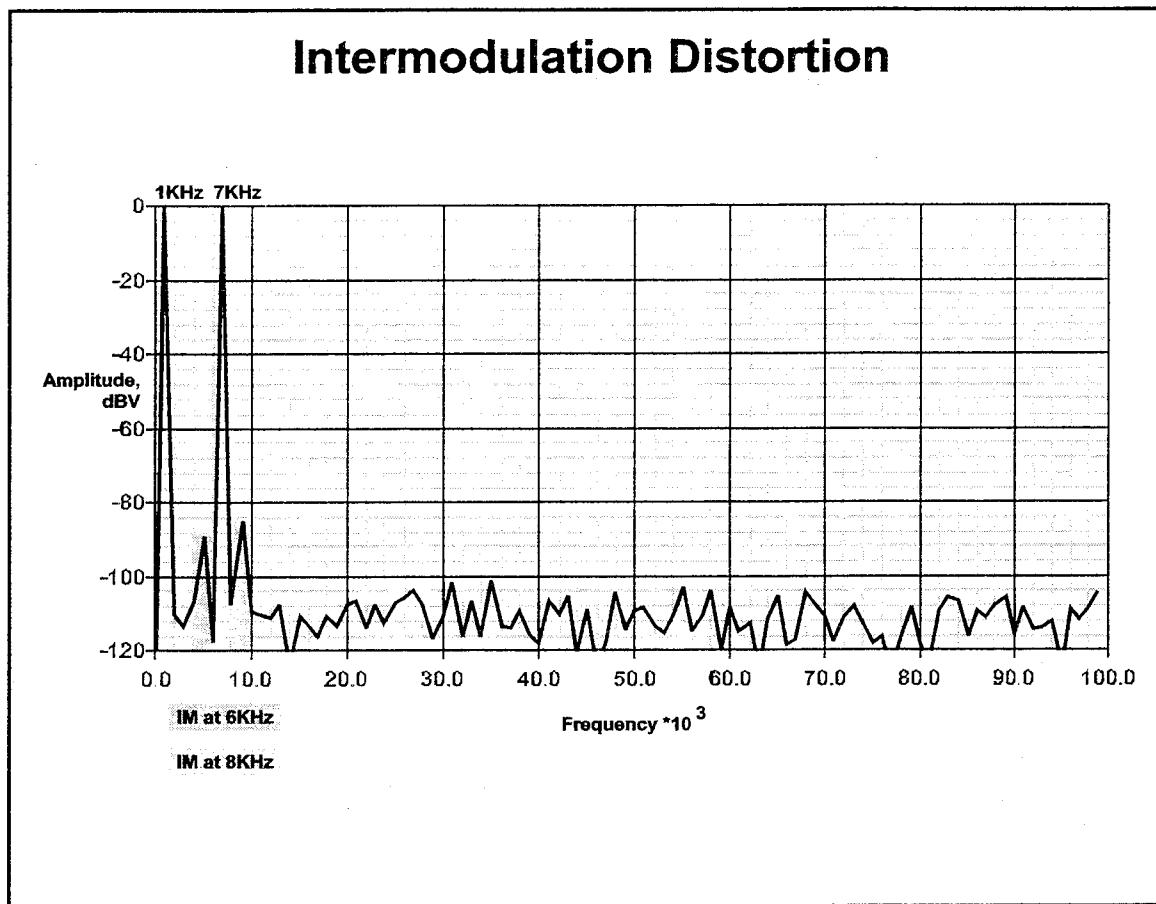


Figure 7.3

NOTES:

Calculating Intermodulation Distortion (IM)

IM is an acronym for *Intermodulation Distortion*, in which a tone containing 2 frequency components is generated with the DAC under test. Testing for IM is a technique which highlights non-linear distortion. (Non-linear in this context does not refer to mathematical functions but to a situation where the principle of superposition does not hold.)

Without going through the math⁵, a non-linearity in the transfer characteristic generates harmonics called *IM products* which are second order distortion (the sum and difference of the 2 sinusoids in the tone, i.e. $f_{t1} + f_{t2}$ and $f_{t1} - f_{t2}$), third order distortion ($2f_{t1} + f_{t2}$, $2f_{t1} - f_{t2}$, $f_{t1} + 2f_{t2}$, $f_{t1} - 2f_{t2}$) etc. for higher and higher frequencies. It is generally accepted that IM products above 5th order can be neglected. The 2 frequency components may or may not be the same amplitude. An example of a nonlinear system resulting in an output that is more than the sum of the 2 input sinusoids is given in reference 6, listed at the end of the chapter.⁶

Calculation of IM distortion is the ratio of the RMS sum of the 2 distortion components divided by the amplitude of the lower frequency (and usually larger amplitude) component.

Generating a coherent 2 sine tone

Recall that coherent sampling is required to avoid spectral leakage. Therefore, both sinusoidal components used in the IM test of the tone must use the same Fs because the input code sequence vectors to the WD create both tone components. To coherently generate the tone, use these steps:

1. Realize that $F_s / N = F_t / M$ still applies to both sinusoidal tone components. Use the equation separately for each frequency in the tone, e.g. $F_{s1} / N = F_{t1} / M_1$ and $F_{s2} / N = F_{t2} / M_2$.
2. Sample count N must be the same for both tone frequencies because there is only 1 signal (containing 2 frequencies) which is being sampled. Select an appropriate sample count N to use in the equations for frequency 1 and frequency 2.
3. Fs must also be the same for both tone frequencies.
4. Select one of the Ft frequencies and choose a ratio I_1 / I_2 of 2 integers for the relationship between it and the 2nd frequency. This will become the ratio of M_1 to M_2 .
5. Calculate an Fs and M for F_{t1}
6. Calculate $F_{t2} = F_{t1} * I_2 / I_1$
7. Make certain that the calculated values match the DUT and test system constraints for sample period resolution. If not, recalculate.

NOTES:



Example: The DUT has a settling time of 200nsec, yielding maximum $F_s = 5\text{MHz}$. We wish to test IM distortion with a 2 frequency tone, with the amplitudes at maximum and the frequencies at 1KHz and 7KHz. Choose $N = 1024$ and $M_1 = 1$. $F_s / N = F_{t_1} / M_1$ gives $5 \times 10^6 / 1024 = 1000 / M_1$ yielding $M_1 = 0.2048$, clearly not an acceptable integer value.

Rearranging the equation to solve for M_1 gives $M_1 = N * F_{t_1} / F_s$; to raise the calculated M_1 value we can decrease F_s or increase N . Setting $M_1 = 1$ is an easy way to calculate a new lower F_s : $F_s = N * F_{t_1} = 1024 * 1000 = 1.024\text{MHz}$.

M_2 can now be calculated as $M_2 = N * F_{t_2} / F_s = 1024 * 7000 / 1.024 \times 10^6 = 7$.

Our test system must be capable of delivering the digital code vectors to the DAC input at a rate of 1.024MHz; if it cannot, choose a valid F_s close to 1.024MHz and calculate backwards to get the actual (exact) F_{t_1} and F_{t_2} frequencies produced by the new F_s . To see an example of this type of tone, open the DSP Lab software, press the *WG from Inverse IFFT* button, set $F_s = 1.024\text{e}6$, $N = 1024$, $M = 1$ and place a frequency component of 0dB in bin 1 and bin 7. Change to the *Oscilloscope* to view the resulting 1KHz + 7KHz tone.

NOTES:



Other parameters needed for dynamic testing

Other parameters to be measured in a practical test situation that are not specifically dynamic parameters are the zero scale output and the full scale range. Maximum conversion rate and settling time are parameters that must be taken into account when setting up the DAC to create an analog signal.

Zero scale output

The measured DAC output value when the zero or null level digital input code is presented to the device.

Full scale range (FSR)

The range between the zero scale and full scale DAC output values. This parameter is measured by taking the difference of the zero scale output and the full scale output of the DAC under test. It is different for each DUT and represents (in a DC sense) the dynamic range of a DAC. The nominal FSR is given in the specification as *Output Range*, and the difference between *Output Range* and the measured FSR is known as *Gain Error*.

Maximum conversion rate

This parameter is self explanatory, and should be provided by the device specification. As a DAC's input changes, its output must be given time to reach the output level and settle. The value of this parameter should include the worst case rate, which is most likely the inverse of the time required to change from zero scale to full scale output.

Settling time

Settling time is the time required for the output to reach and remain constant within $\pm 1\%$ LSB of its final value or within some other defined limit. It may be expressed in LSBs or in %FSR.

NOTES:



Testing DAC Dynamic Parameters

Before discussing how to test DAC dynamic parameters, let's review these things:

- ❖ A general test system configuration for DAC dynamic parameter testing
- ❖ A DAC specification
- ❖ A list of DAC dynamic parameters and the information required to calculate their values
- ❖ A list of required test system items
- ❖ A list of steps to do the tests

NOTES:

Test System Configuration for DAC Dynamic Parameter Tests

As you can see in Figure 7.4, the WD and DSP components of a mixed signal test system are key elements in testing dynamic parameters. The basic approach to testing Signal to Noise Ratio (SNR), Total Harmonic Distortion (THD), etc. is to have the DAC create as pure a sinusoidal wave as possible, then analyze it to see how pure it really is. Requirements on the digital subsection of a test system are more stringent for testing dynamic parameters than static parameters. Coherent sampling requires synchronized digital and analog subsystems.

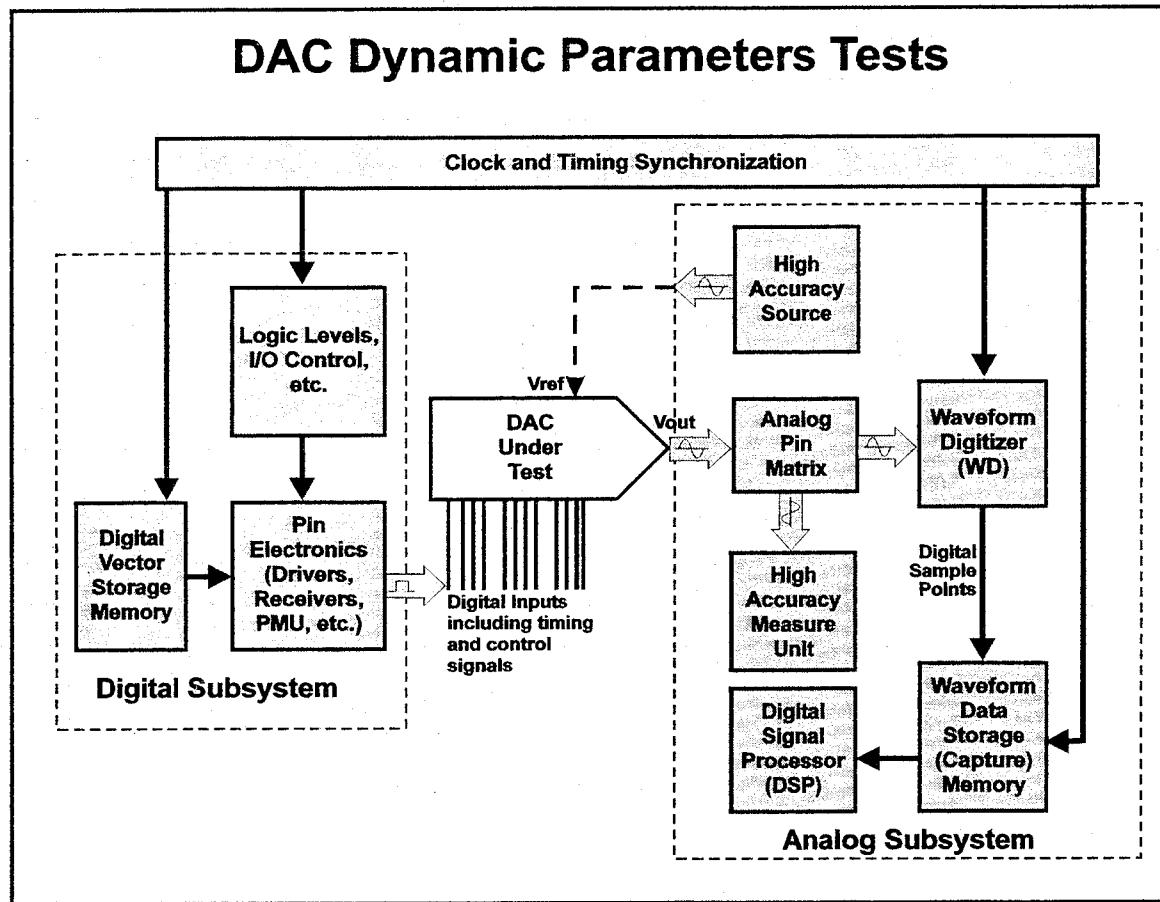


Figure 7.4

NOTES:



Example DAC Data Sheet

The following table contains an example DAC data sheet; notice the dynamic parameters—signal to noise ratio (SNR), total harmonic distortion (THD) and intermodulation distortion (IM). Note that the limit values for the dynamic specs are in dB. A parameter to be discussed that is not in the specification is SINAD, a ratio of the signal to total noise and distortion.

Parameter	Conditions	Min	Typ	Max	Units
INPUT					
Resolution		12			Bits
Input format	straight binary				
STATIC					
Offset error	input = 0x0			±0.2	% FSR
Gain error	input = 0xFFFF			±0.4	% FSR
Differential nonlinearity	guaranteed monotonic to 12 bits			±1	LSB
Integral nonlinearity				±½	LSB
Output range			±2.5		V
DYNAMIC					
SNR	f _{out} = 1000Hz sinusoid	68	70		dB
THD	f _{out} = 1000Hz sinusoid, f _{max} =20KHz		-69	-67	dB
IM	f _{out} = 1000Hz + 7000Hz tone			-65	dB
AC					
t _{settle}	Max input change = 16 LSBs		180	200	nsec
Conversion rate				5	MHz

Table 7.1 DAC Specifications

NOTES:

Digital to Analog Converter Dynamic Parameters

Parameter Measurement Requirements

Measuring dynamic parameters requires coordination of a number of test system and DUT elements. Fortunately, once everything is set up properly, one set of sample points will provide the data for several parameter calculations. Following is a table showing the measurement requirements for SINAD, SNR, THD and IM.

Parameter to test	Measurement(s) required	Items needed for measurement(s)
SINAD, SNR, THD	<ol style="list-style-type: none"> 1. Zero scale 2. Full scale 3. A set of sample points of DAC output 4. DSP frequency analysis 	<ul style="list-style-type: none"> ❖ Digital sine wave input samples (requires DAC resolution, F_t, F_s, M, N and calculation method)
IM	<ol style="list-style-type: none"> 1. Zero scale 2. Full scale 3. A set of sample points of DAC output 4. DSP frequency analysis 	<ul style="list-style-type: none"> ❖ Digital codes for tone (sum of 2 sine) wave input samples (requires DAC resolution, F_t, F_s, M, N and calculation method)

Table 7.2 Measurement requirements for dynamic parameter testing

NOTES:

Items and Steps Required to Dynamically Test a DAC

Many different software, algorithmic and hardware items are needed to do a dynamic test on a DAC. The following items will be discussed in this chapter:

1. digital input codes (vectors) must be created and stored for use by the test program
2. F_t , F_s , M and N must be set up to reflect the specification requirements
3. a smoothing filter for DUT output is needed to remove the "steps" from a sine wave
4. an anti-alias filter is needed to filter DUT output before digitizing it (may be same filter as smoothing filter)
5. a notch filter may be needed to remove the fundamental without removing any harmonics or noise of interest
6. amplification or attenuation may be needed so the DUT output signal fits within the WD input range
7. a level shifter may be needed so the DUT output signal fits within the WD input range
8. setup of the WD on the test system, which requires its own F_s , M and N based on the DAC output signal F_t .

Keep in mind that items 4 through 7 above and 3 through 9 below also apply to dynamically testing any analog signal. When everything is properly set up to test the DAC, the steps required to perform dynamic tests are as follows:

1. Set up coherent test conditions for coherent DAC output samples, i.e. F_t , F_s , M and N .
2. Make a continuous output signal from the DAC by sending digital codes to DAC at F_s rate
3. Coherently collect a set of output samples with the waveform digitizer (with fundamental removed).
4. Send the collected set of time samples to the DSP to perform DFT/FFT analysis
5. If necessary, convert rectangular frequency results to polar (magnitude and phase) results
6. Compensate results for signal conditioning done on signal
7. Put fundamental into bin M because it was removed
8. Analyze the frequency bins of interest using equations or tester algorithms for SINAD, SNR, THD and compare to specification.
9. Make a pass/fail decision based on the results.

NOTES:

Creating the DAC Output Signal

Calculating the digital input signal as an array of points

To measure dynamic parameters such as SINAD, THD and SNR, a sinusoidal wave must be created by the DUT and measured by the test system. The purity of the output wave is evaluated by the various tests, where specific parameters define specific impurities in the sine wave. The first goal is to create a digital input which will produce a sine wave at the output of the DAC under test.

Our DAC has a limited number of discrete output points which depend on its resolution; an output wave is created by setting DAC input codes which match, as closely as possible, points on a sine wave. A table of digital codes must be generated to represent the input sine wave; the codes in this table are sent to the DAC input at the sample rate F_s which creates the desired output sine wave frequency. The sine wave sample points can be generated using an inverse DFT/FFT as discussed in Lab Exercise number 6.3 in chapter 6 or they can be generated by using a sine wave equation as discussed here.

Using a sine wave equation

Recall that all these dynamic tests are a ratio of output error components to the generated signal and that we wish to test how good the DUT can be. Maximizing DUT *goodness* requires the generated signal to be as large as possible, so we want the digital codes to the DAC input to go as close as possible to full scale. Given that points on a sine wave can be calculated with the equation

$$Y = A \sin(\omega t + \phi) \quad (7.1)$$

where Y is the point on the sine wave, A = maximum amplitude, ω = angular frequency ($2\pi F_t$) in radians, t = time and θ = phase shift in radians or degrees. Given the DAC specifications, we can use equation (7.1) to calculate the input codes to generate our sine wave.

The requirements for DAC output samples are:

1. The sample rate must not exceed the DAC maximum conversion rate (or 1 / settling time).
2. (*This applies to coherent sample generation only*) The created points must form an exact sinusoidal cycle or cycles so it repeats correctly. This means there can be no partial sine wave periods in the sample set.

NOTES:

The portion of the specification for dynamic tests is duplicated below. Refer to it for the following discussion.

DYNAMIC					
SNR	$f_{out} = 1000\text{Hz}$ sinusoid	68	70		dB
THD	$f_{out} = 1000\text{Hz}$ sinusoid, $f_{max}=20\text{KHz}$		-69	-67	dB
IM	$f_{out} = 1000\text{Hz} + 3100\text{Hz}$ tone			-65	dB
AC					
t_{settle}	Max input change = 16 LSBs		180	200	nsec
Conversion rate				5	MHz

Table 7.3

Given the condition of $f_{out} = 1000\text{Hz}$, equation (7.1) becomes, with phase shift = 0,

$$Y = \sin(2\pi 1000t) \quad (7.2)$$

Note that the output amplitude specified as "Output range" = $\pm 2.5\text{V}$ is not included in the equation because we will be calculating values relative to the 12 bit input word not relative to the output voltage swing. When a set of codes covering its full scale input range is delivered to the DAC, it produces its full scale output swing, whatever that may be.

The values selected for t must occur at a sample time interval which allows the DAC to settle to its final value and which produces an adequate number of samples and has a test frequency as close as possible to the desired output frequency of 1000Hz.

Coherent sample generation uses the coherency requirements relating sample frequency F_s , test frequency F_t , number of samples N and number of cycles M with the familiar equation:

$$\frac{F_s}{N} = \frac{F_t}{M} \quad (7.3)$$

Note that the sample points being generated *will not* be used in an FFT algorithm, only sent to the DAC under test, so there is no requirement for 2^N sample points. We know $F_t = 1000\text{Hz}$; to create the maximum samples per cycle (and the fastest test time), choose $M = 1$, i.e. all samples will occur within one cycle. Given the settling time specification of 200nsec, the input data can change at a 5MHz maximum rate (1/200nsec) and still have

NOTES:



the output value settle to its final value. Thus the minimum time between samples is 200nsec. This satisfies the first requirement that the inverse of the sample rate must not exceed the settling time.

The sample period of 200nsec divides evenly into 1msec ($1/1000\text{Hz}$) so the second requirement is also satisfied—there is no fractional portion of a 1000Hz waveform created by making 5000 samples at a rate of 1 every 200nsec because $5000 \times 200\text{nsec} = 1\text{msec}$ exactly.

Recall that this is a 12 bit DAC which has only 4096 points in its transfer line. Thus the 5000 codes generated to create the 1000Hz output wave will contain duplicates and will not contain every unique code of the 4096 possible codes. (All possible codes create a straight line not a sinusoid.) This is ok; the important aspect of the input codes is that they are as close as possible to a value on the theoretical sine curve so that any distortion due to input data is near the theoretical minimum quantization error.

Given equation (7.2) and equation (7.3), how do we relate them to create a table of sine wave points which can be fed to our 12 bit DAC? First, we need to establish all values of t in the sine equation as a set of points on the time axis at which the DAC will put out a sample. We established earlier that $M = 1$, $N = 5000$, $F_s = 5\text{MHz}$ and $Ft = 1000\text{Hz}$. What we want are the amplitude values Y of a sine wave at every sample point for the 5000 samples. Seen in table format, it is:

t	Y	Y in 12 bit decimal	Y in 12 bit hexadecimal
0nsec	0	2048	0x800
200nsec	0.0013	2051	0x803
400nsec	0.0025	2053	0x805
600nsec	0.0038	2056	0x808
800nsec	0.0050	2058	0x80A
1000nsec	0.0063	2061	0x80C
1200nsec	0.0075	2063	0x80E
etc.			

Table 7.4

Notice that the values for **Y in 12 bit decimal** begin at 2048 rather than 0; recall that our DAC under test has "straight binary input" which, for 12 bits, requires input codes in the range of 0 to 4095. The sine wave must go positive and negative by equal amounts, which requires that we start in the center of the code range: 2048. The 12 bit value must be mathematically offset by half scale to do this, as seen in the following computer algorithm. It will create a set of points for any resolution DAC; it is written in C but can be easily translated to Pascal or Basic:

NOTES:

```

int BitFields = 0; /* Bit mask to remove high bits from the final DAC code */
int Index, DUTinputY;
int DUT_Bits = 12; /* Resolution of our DAC */
double Y, t, InputFSR;
double Fs = 5.0e6; /* Sample rate as set by DAC settling time */
double TwoPi = 8.0 * atan(1); /* Calculate  $2\pi$  using arctan */
for (Index = 0; Index < DUT_Bits; Index++)
    BitFields = (BitFields << 1) + 1; /* Put 12 ones in bit mask */
InputFSR = pow(2.0, (double)DUT_Bits) - 1; /* 4095 for 12 bit DAC */
for (Index = 0; Index < N; Index++)
{
    t = Index / Fs;
    Y = (1 + sin(TwoPi * Ft * t)) / 2;
    DUTinputY[Index] = (int)(Y * InputFSR) & BitFields;
}

```

Note that t , Y , Fs , Ft and InputFSR are *real* variable types, Index , N , DUT_Bits are integers and BitFields is an unsigned integer. All integers should be at least 16 bits in length, and should be long (24 or 32 bit) integers if the DAC could have 16 or more input bits. Line by line, the algorithm does the following:

Declare and initialize variables as necessary:

```

int BitFields = 0; /* Bit mask to remove high bits from the final DAC code */
int Index, DUTinputY;
int DUT_Bits = 12; /* Resolution of our DAC */
double Y, t, InputFSR;
double Fs = 5e6; /* Sample rate as set by DAC settling time */
double TwoPi = 8 * atan(1); /* Calculate  $2 \cdot \pi$  using arctan function */

```

Fill each bit in the BitFields variable with 1's up to the number of bits in the DUT (12 for our example) by left shifting and adding 1:

```

for (Index = 0; Index < DUT_Bits; Index++)
    BitFields = (BitFields << 1) + 1; /* Put 12 ones in bit mask */

```

Calculate the full scale value which the DUT can have, as a function of its input bit resolution, as $2^{\text{DUT_Bits}} - 1$:

```

InputFSR = pow(2.0, (double)DUT_Bits) - 1; /* 4095 for 12 bit DAC */

```

Execute a loop N times to calculate the time and amplitude for each sample point:

```

for (Index = 0; Index < N; Index++)
{

```

NOTES:

Digital to Analog Converter Dynamic Parameters

Calculate the x axis time coordinate for this sample:

$$t = \text{Index} / \text{Fs};$$

Calculate the y axis amplitude coordinate for this x axis point t , offset by 1 so the result ranges from [0...2], then divide by 2 to bring the range to [0...1]. TwoPI was precalculated in the initialization section earlier.

$$Y = (1 + \sin(\text{TwoPi} * \text{Ft} * t)) / 2.0;$$

Multiply the calculated Y value by the maximum DAC input value and set all bits except the lowest 12 (number of input bits) to zero with a bitwise AND operation. Store the result in an array:

$$\text{DUTinputY}[\text{Index}] = (\text{int})(Y * \text{InputFSR}) \& \text{BitFields};$$

Terminate the loop:

}

Note that the algorithm can be made more efficient by moving the divide by 2 out of the for loop and putting it in the InputFSR calculation; it was written in the less efficient way so it is easier to understand. If the DAC has *offset binary*, *sign magnitude binary* or other input format, the algorithm must be changed appropriately.

NOTES:



Lab Exercise 7.1—Creating DAC inputs for a sine wave

Lab goals

- ◊ Set the parameters required to create the DAC digital input codes for a sine wave
- ◊ Create and view the output wave of a DAC
- ◊ View the input sample points creating the DAC output
- ◊ Look at the effects of DAC resolution, sample count N and cycle count M

Lab objectives

- ◊ Understand how to create a sine wave for a DAC under test
- ◊ Understand the effect of coherent sampling equation parameters on DAC input codes
- ◊ Recognize that the created sample points reflect the time characteristics of a sine wave
- ◊ Recognize that the output wave from a DAC is only as good as the input data

This lab uses the algorithm given on page 229 to generate a set of digital input codes for the DAC specified in Table 7.1 on page 223 so it will make a sine wave that matches the test conditions for the dynamic specifications, which are repeated here:

Parameter	Conditions	Min	Typ	Max	Units
SNR	$f_{out} = 1000\text{Hz}$ sinusoid	68	70		dB
THD	$f_{out} = 1000\text{Hz}$ sinusoid, $f_{max} = 20\text{KHz}$		-69	-67	dB

Table 7.5

In this exercise you will make the set of points required to generate a sine wave to meet the *Conditions* specified in Table 7.5. Note that the test conditions for both specifications require a 1000Hz sinusoid as the output frequency.

Open the DSP Lab software application and press the *WG from Equation* button. An oscilloscope (time domain) graph exists to show DAC output sample points for a sine wave. No points are drawn until all entry boxes are filled. The DAC input code is shown on the right vertical axis; the equivalent output signal for the DAC, nor-

NOTES:

normalized to $\pm 1V$, is shown on the left vertical axis. You see an entry box for *Bits*; the *Sampler* button allows entry of *Ft*, *Fs*, *N* and *M*.

- ❖ This is a 12 bit DAC so type 12 into the *Bits* box.
- ❖ Open the *Sampler* and set it to *Calculate Ft*.
- ❖ Set *Fs* to the maximum DAC output update frequency of 5MHz based on the 200nsec settling time (type 5E6 or 5000000)
- ❖ Set *N* to 5000. (Refer to the discussion starting on page 227 if necessary.)
- ❖ This calculation was based on a single cycle so set *M* to 1 and press *Enter*.

Lab Question 7.1.1: What is the calculated value of *Ft*?

Close the *Sampler* and look at the *x* axis with *Sample Number*.

Lab Question 7.1.2: **a)** How many points are drawn for the single waveform cycle? **b)** From the graph alone, can you tell what time increment is associated with each sample point? **c)** What determines the time increment between each sample point?

Look at the right side *y* axis with *DAC Input Code*.

Lab Question 7.1.3: **a)** What are the minimum, maximum and "waveform starting point" codes? **b)** What sets the maximum *DAC Input Code*?

Click *Show Steps* so that it is checked. Click again so it is not checked.

Lab Question 7.1.4: Can you see any difference in the displayed sine wave? Why or why not?

NOTES:

Press the *View Points* button. Examine the list which shows *Index*, *Time*, *Amplitude* and *DAC Input*.

Lab Question 7.1.5: The first *Index* value is 0; what will the last value be? (Press the End key to see.)

Lab Question 7.1.6: What is the time increment between each sample index value?

Lab Question 7.1.7: a) What is the initial value of *Amplitude*? **b)** How does it relate to the initial value of *DAC Input*?

The maximum value of *Amplitude* for the sine wave is 1. Page down (PgDn key) the list from the beginning and find the maximum value of *DAC Input*.

Lab Question 7.1.8: a) What is the maximum value? **b)** Does the maximum value exist for more than one *Index* in a row? **c)** Do other *DAC Input* values sequentially repeat? **d)** Why? **e)** What does this indicate about the output wave?

Page down again until *Amplitude* crosses 0. This is the mid-scale *DAC Input* value.

Lab Question 7.1.9: a) What is the mid-scale *DAC Input* value? **b)** Does it sequentially exist for more than one *Index*? **c)** Do other values in this area of codes sequentially repeat? **d)** Why?

Close the *Sine Generator Points* dialog by pressing the Enter or Escape key. Set the *Show Steps* box so it is **not** checked. Open the *Sampler* and change *N* from 5000 to 50 and press *OK*.

NOTES:

Digital to Analog Converter Dynamic Parameters

Lab Question 7.1.10: a) What happened to the value of Ft and the maximum x axis *Sample Number* value? b) Does the wave look different?

Set the *Show Steps* box so it is checked.

Lab Question 7.1.11: a) Does the wave look different now? b) Why or why not? c) Does the "stepped" or "not stepped" waveform more closely approximate the output of an actual DAC?

Press the *View Points* button. Examine the new list of points.

Lab Question 7.1.12: a) How many points are in this list? b) Are there any sequential duplicate *DAC Input* values? c) What are the maximum and minimum *DAC Input* values? d) Do they reach the zero and full scale input code values for a 12 bit DAC?

Close the *Sine Generator Points* dialog by pressing the Enter or Escape key. Open the *Sampler*, change *M* to 3 and press *OK*.

Lab Question 7.1.13: a) What happened to Ft ? b) How many cycles are generated for the number of *N* points? c) With *Show Steps* checked, are the vertical step sizes the same for all steps? d) Why or why not?

Open the *Sampler*, change *M* to 29 and press *OK*.

NOTES:



Lab Question 7.1.14: a) What is Ft ? b) Is it less than half Fs ? c) Is $N = 50$ enough points to uniquely characterize a sine wave at the calculated Ft frequency? Change N back to 5000 and answer these questions again.

Set *Show Steps* so it is **not** checked and *Show Samples* so it **is** checked; change M to 1 and make a note of Ft . Then change *Bits* to 5.

Lab Question 7.1.15: a) What effect does changing *Bits* have on Ft ? b) What is the maximum *DAC Input Code*? c) How does the wave look? Notice how many duplicate points are at the same step. Press the *View Points* button and see this in the list of values. d) Why is this?

Experiment with the parameters on the Sine Generator to see the interactions among them. Especially change *Bits* to 14 and 16 and *View Points* to see 16 bit data. If you create a set of points that you wish to use in a test program, you can save the points to a text file with the *Sine Generator Points* dialog with the *Save Points* button.

End of Lab Exercise 7.1.

What to do with the list of codes

Two methods have been presented which will generate a list of DAC input codes so the DAC under test will produce a (stepped) sine wave. (One using the IFFT was in Lab Exercise 6.3 on page 200.) We now have a list of values (N values in the list) which can be sent to the DAC under test that will create some M number of cycles of a sinusoid. What can be done with the values in the list? As binary values, they are essentially a set of digital input vectors which should be sent to the DAC inputs at a clock frequency equal to the sample frequency Fs . The set of vectors should be repeated continuously for the duration of a measurement to create a continuous sinusoid out of the DAC at frequency Ft .

Filtering the output signal

The DAC output changes in discrete steps which are at least 1 LSB in amplitude and possibly more. This waveform is distorted by the steps. To allow the wave to be sampled for distortion, we must filter out the steps to

NOTES:

Digital to Analog Converter Dynamic Parameters

create a smooth continuous sinusoid. The step frequency which must be filtered is F_s , the sample frequency. The higher the DAC update frequency F_s as compared to the test frequency F_t , the easier it is to filter out the steps. Look at Figure 7.5, a frequency plot of a sine wave with one harmonic and one F_s frequency component.

A low pass filter which removes the F_s component is required. The primary requirements are a flat pass band and a sharp roll-off above the maximum harmonic frequency but below the sample frequency. Note that the filter must be an analog filter—a switched capacitor filter will add its own high frequency component(s). Often a DAC specification will specify a filter schematic. For our DUT, a low pass filter which rolls off at 25KHz will smooth the output signal without excluding frequencies contained below the f_{max} condition of 20KHz specified for SNR and THD tests.

NOTES:

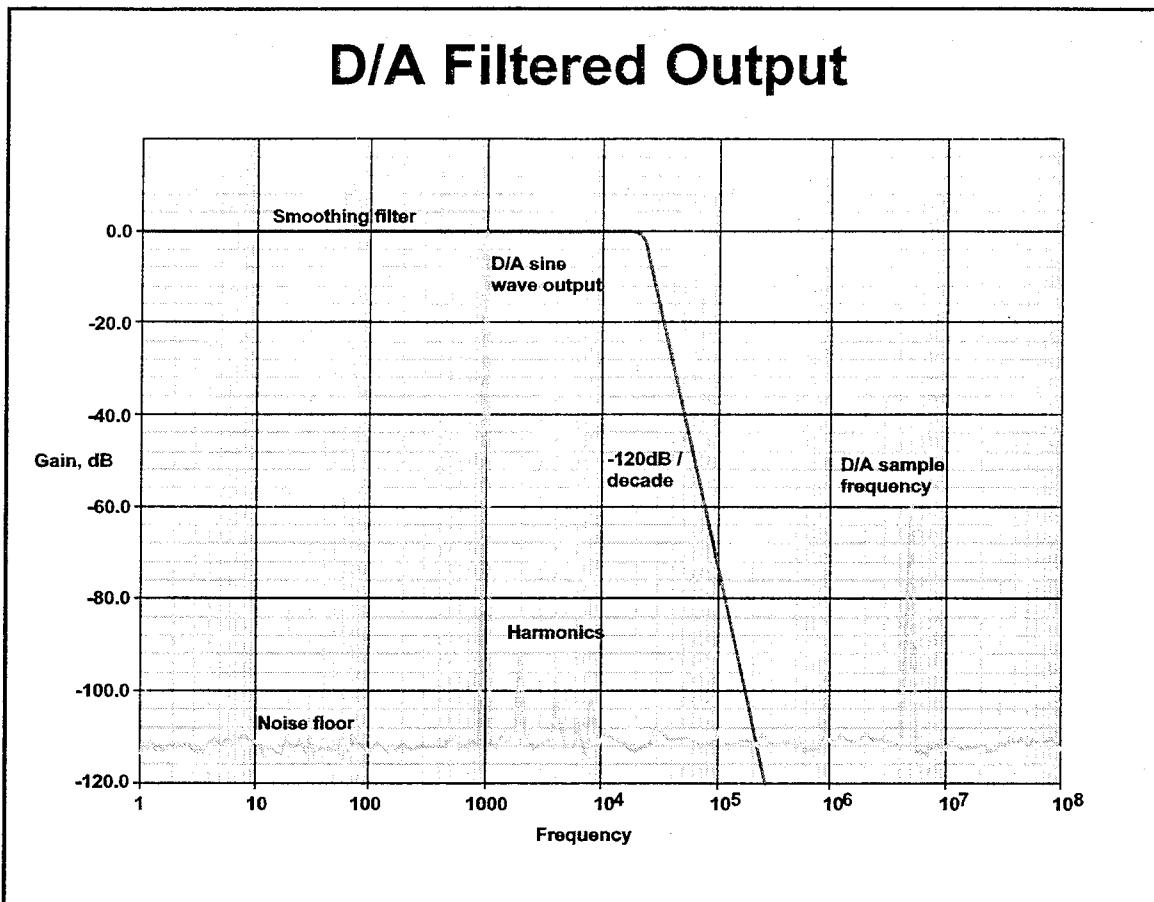


Figure 7.5

Notice that the roll-off is very steep, at -120dB per decade, indicating a 6 pole filter. The filter must be carefully designed or selected so that it does not add its own distortion to the analog signal. Mixed signal test systems often have sharp cutoff anti-aliasing filters which can be used as a low-pass signal filter (while simultaneously performing the anti-alias function).

Question 7.1: What is the number of the highest harmonic that can be measured without attenuation using the filter shown in Figure 7.5?

NOTES:

Digital to Analog Converter Dynamic Parameters

We have created a "smooth" sine wave with the DAC under test. The next task is to digitize the wave and analyze the digitized results to calculate measurements which are compared with the specifications.

NOTES:

Capturing the DAC output

Special Note: Almost everything from this point to the end of the chapter applies to digitizing any analog sine wave and analyzing it for distortion. It can be applied to filters, amplifiers or any other component with a sinusoidal analog output, not just DACs.

Digitizing the filtered continuous sinusoid we have created allows us to use the test system's available DSP tools to calculate the frequency related parameters such as signal to noise ratio and harmonic distortion. There are some considerations and trade-offs to make so that the digitized information is useful. Look at the block diagram of the waveform digitizer in Figure 7.6 (repeated from chapter 1):

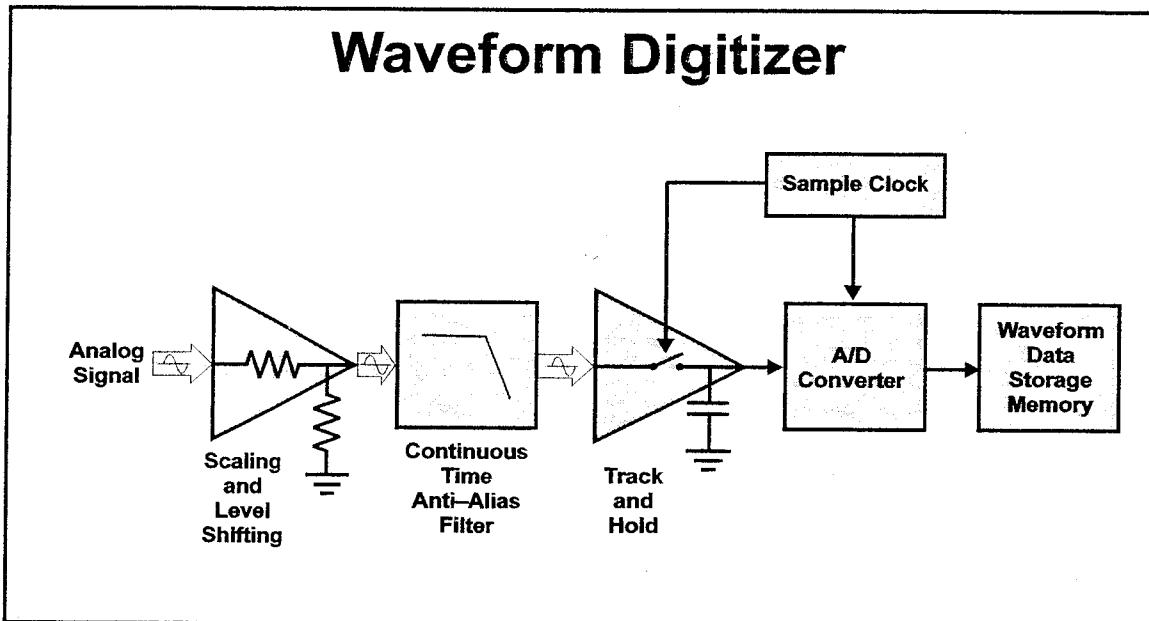


Figure 7.6

NOTES:



Conditioning the analog signal for the waveform digitizer

The block labelled "Scaling and Level Shifting" represents conditioning for the incoming analog signal generated by the DAC under test. A signal may require conditioning for many possible reasons:

- ❖ The analog signal range is too small or too large for the waveform digitizer to process
- ❖ The analog signal is referenced to a voltage level which is different from the test system analog reference level (for example, *DUT ground is different from tester ground*)
- ❖ The analog signal is *differential* and must be converted to *single-ended*
- ❖ The analog signal is a current and the waveform digitizer requires a voltage input
- ❖ The DUT output has a (relatively) high impedance and the waveform digitizer disturbs the analog signal during the digitizing process

These represent amplitude, zero reference and impedance conditioning; depending on the DUT there may also be requirements for frequency conditioning, demodulation, etc. before the DUT signal is digitized. For our DAC under test, the output is a low impedance single ended voltage waveform which is referenced to a DUT analog ground which we have connected to the tester analog zero reference point.

Before we can proceed, we must know the specifications of the test system waveform digitizer. Assume our test system has a digitizer with the specifications given in Table 7.6:

Parameter	Min	Max	Units
Resolution		14	Bits
Sample rate range	100	1 000 000	Hz
Sample rate resolution	10		Hz
Accuracy		0.1	% FSR
Input voltage range		0 to +5	V
Signal to noise and distortion	70		dB
Total harmonic distortion		-72	dB

Table 7.6

At first glance, it is easy to see that we cannot *directly* measure our DUT's THD and SNR with this digitizer because the digitizer is no better than the DAC under test! The input voltage range is given as 0 to +5V, so the DAC's -2.5 to +2.5V output must be DC shifted by +2.5V. We must presume that the digitizer has differential inputs to allow negative voltage in, otherwise we will have to level shift our signal before passing it to the digitizer. With a resolution of 14 bits and FSR of 5V, an LSB size calculates to 305µV. The sample rate range is from 100Hz to 1MHz, in resolution increments of 10Hz.

NOTES:

Question 7.2: Given a DAC under test with FSR of $\pm 2.5\text{V}$ with a maximum THD specification of -67dB and the equation for THD:

$$-67 = 20\log\left(\frac{V_{dist}}{\left(\frac{2.5}{\sqrt{2}}\right)}\right) \quad (7.4)$$

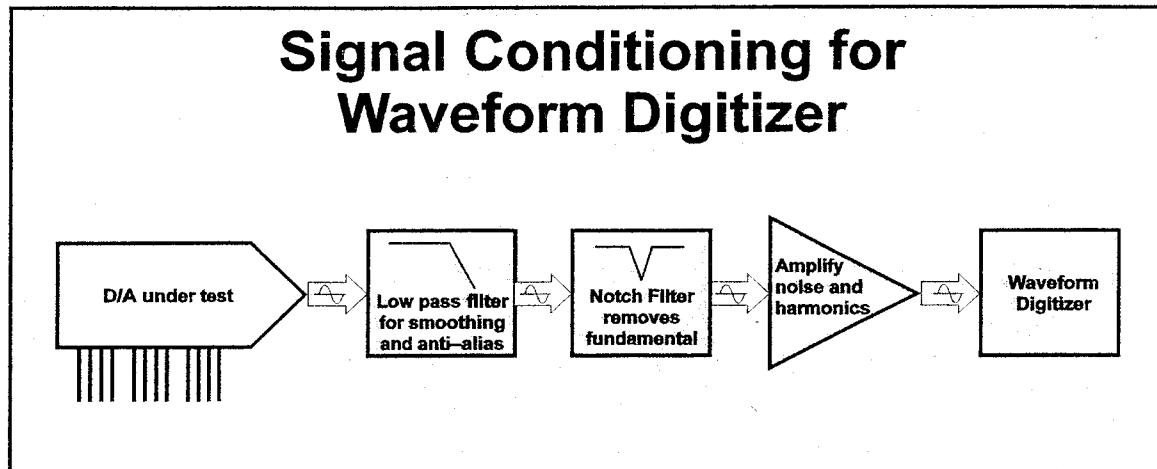
(The $\sqrt{2}$ factor converts peak to rms.)

- a) What is the voltage level for a signal (V_{dist}) that is -67dB below the DUT FSR? (Recall that this voltage represents the sum of all harmonic distortion components for the DUT.) b) Given that 10 harmonics make up the sum, what is the average voltage (V_{dist}) for each single harmonic?
-
-
-

What measurement resolution is required to adequately digitize these low voltage levels? A generally accepted rule is that the measurement system must be 10 times better than what is being measured, so *good enough* measurement resolution is on the order of tens of μV or less.

As we discovered from the waveform digitizer specification, it has an LSB size of $305\mu\text{V}$, not quite *good enough* by a factor of *at least* 2. How can we measure $79\mu\text{V}$ (the answer to Question 7.2) with this waveform digitizer? It's like trying to measure your height with an automobile odometer. DACs are tested on automatic test equipment all the time...you think there must be some way to do it and you are right—use another filter and amplification.

NOTES:

**Figure 7.7**

An additional filter is used to filter out (actually to attenuate) the fundamental as seen in Figure 7.7 and Figure 7.8. The *notch* filter attenuates the main component of the analog signal created with the DUT.¹ The remaining signal can be amplified to boost the noise and harmonics so the waveform digitizer resolution is good enough. With notch filter attenuation of 40dB as shown in Figure 7.8, the full scale fundamental will be attenuated by a factor of 100. For our example, a factor of 100 to 1000 amplification is sufficient. A value of 100 is chosen as the maximum boost to keep the signal in the input range of the WD, changing $79\mu V$ into $7.9mV$. This gives an equivalent WD to DUT resolution ratio of approximately 26, more than the required 10 times better. The value of 26 comes from the ratio of $7.9mV / 305\mu V$.

Question 7.3: Is it practical to use a low pass filter to remove the fundamental instead of a notch filter? Why or why not?

NOTES:

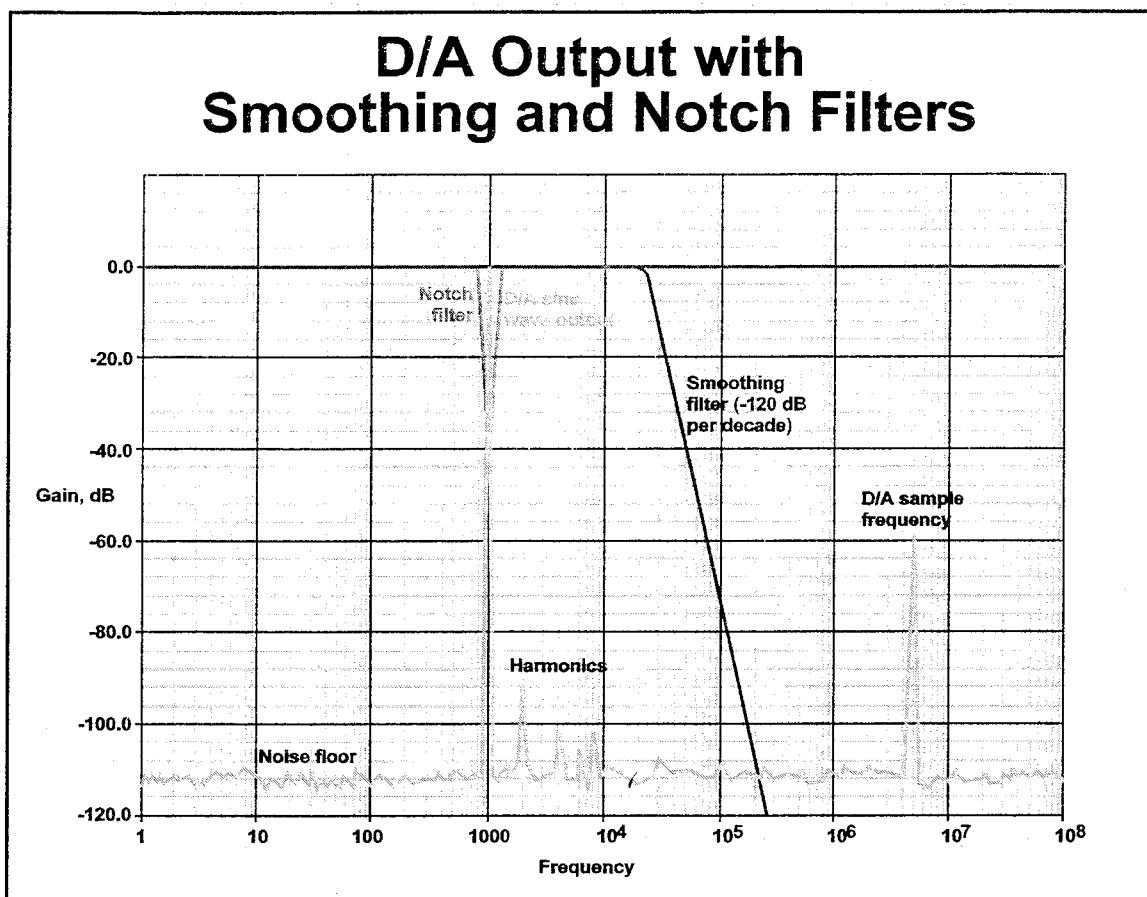


Figure 7.8

Naturally, the factor of amplification must be taken into account when calculating parameters such as SNR or THD. Also, the amplification factor must be as exact as possible—if an op amp is used as the amplifier, use 0.1% accurate resistors (e.g. Vishay) and a low temperature coefficient trimpot to set the gain. If possible, set up an autocalibration system with a reference sine wave source to store values for amplification and filter attenuation. Use the stored values in parameter calculations prior to testing devices.

Also, use a low distortion op amp and high quality capacitors for the gain amplifiers and filters (if you build your own filters). Be aware of the gain-bandwidth of the op amps and be certain that, at the gain you set, the

NOTES:

Digital to Analog Converter Dynamic Parameters

amplifier remains in its flat gain region at the highest harmonic you wish to measure (*not just at the fundamental frequency!*).

Digitizing the (filtered) analog signal

What remains is to calculate the number of samples we want (N), the sample frequency (F_s) and the number of test frequency cycles (M) over which to take the samples such that the result is a set of coherent samples of a 1KHz fundamental test frequency (F_t)

We removed the fundamental so how do we have an F_t of 1KHz? Even though the fundamental has been attenuated, we must sample the signal over an integer number of F_t cycles so that the noise and harmonics are placed in the correct frequency bins after performing a Fourier transform. In other words, we still want to calculate the harmonic components of a 1KHz waveform.

NOTES:



Lab Exercise 7.2—Real World Example of Digitizing a Sine Wave

Lab goals

- ◊ Set the parameters required to coherently digitize the filtered analog sine wave
- ◊ Select parameter N such that the FFT algorithm can be used
- ◊ Illustrate that coherent sampling requirements may conflict with WD resolution
- ◊ Illustrate some solutions to the conflicting requirements problem
- ◊ Calculate different values of F_s , N and M to optimize them for minimum test time

Lab objectives

- ◊ Review coherent sampling
- ◊ Understand that F_t is the same for both the DAC and the WD
- ◊ Understand that the choice of WD sampling parameters has an effect on test time because of the DFT versus FFT calculation time difference
- ◊ Understand that different combinations of coherent equation parameters are possible and have an effect on F_t accuracy and test time (via UTP)

This lab creates a sine wave with the *Fourier Series Waveform Generator* that simulates the DAC output wave, which has F_t fixed at 1000 as set by the DAC parameters. Start the DSP Lab software and first press the *Calculator* button because it will be needed. Next, return to the lab's button window and press the *Spectrum Analysis* button. In the *Fourier Waveform* tab follow these steps:

1. Press the *Clear Harmonics* button to remove any waveform set previously.
2. Enter 1000 in the *Fundamental Frequency* box.
3. Select *Harmonic Number* 1 and type 1 into the *Peak Amplitude* box; leave *Overall Scale Factor* at 1.
4. Change to *Oscilloscope* and press the *Sampler* button.
5. Set the Sampler to *Calculate F_s* .

Lab Question 7.2.1: Set $N = 1024$ and $M = 3$. What is the value of F_s required to get $F_t = 1000\text{Hz}$ as required by the DAC output signal?

NOTES:

Digital to Analog Converter Dynamic Parameters

Lab Question 7.2.2: a) With the sample rate resolution given in Table 7.6 on page 240, can the WD sample at the F_s rate from Lab Question 7.2.1? b) What is the smallest F_s change that can occur for the WD? c) What is the closest value of F_s that the WD can have?

Lab Question 7.2.3: Select *Calculate Ft* and set $F_s = 341330$. a) What is the value of Ft ? b) Can the specified WD sample at this frequency? c) Is this a valid solution?

Lab Question 7.2.4: By selecting $M = 3$, the time to collect samples is more than with $M = 1$. Set $M = 1$ and a) recalculate the F_s required to get $Ft = 1000$. b) Can the WD sample at this frequency?

Lab Question 7.2.5: With the Sampler "Calculate" button set to calculate F_s , find a combination of F_s , M and N that will work for the specified WD with a 1000Hz Ft and no leakage. Give the parameters Ft , F_s , M and N , and calculate the UTP and $Fres$ for each.

NOTES:



Calculating the result parameters

We have successfully digitized sample points of the DAC output, and this data is now stored in an array in capture memory as a set of samples which represent $N = 1024$ points along the DAC's output sine wave. Next, we wish to extract the desired information from this set of points so it can be compared to the DAC specification parameters.

First, everything we want to know is related to the frequency characteristics of the signal so it is necessary to convert the data to a frequency spectrum by performing an FFT or DFT. With an integer power of 2 points an FFT is the fastest way to go; mixed signal test systems have a digital signal processor (DSP) which will accept this array of points, perform an FFT then return an array of equivalent size containing frequency information. Often, there are also more sophisticated functions which will directly return SNR, THD, etc. calculations.

Any signal conditioning done to the DAC output signal may be compensated in the time samples before sending the time data to the DSP for processing. This includes restoring the fundamental that was removed by the notch filter and compensating for any gain or attenuation of the overall amplitude.

Overall gain correction

The data passed to the DSP is a single dimensioned array which contains only amplitude information (and no time information). Before we can pass the amplitude array to the FFT routine, each amplitude point must be mathematically compensated for the gain used to condition the signal. This is done by dividing each value by the gain factor we selected, which was 100. This can be done using a "for" loop:

```
for (Index = 0; Index < N; Index++)
    Amp1[Index] = Amp1[Index] / 100;
```

Mixed signal test systems usually have a routine which will multiply an array of points by a constant value; if this is available, use it instead of the loop as it will execute faster.

Time to frequency conversion

The time/amplitude data array is now passed to a routine which performs an FFT. The returned information, although we say it is frequency data, really is just another (actually 2) single dimensioned array(s) of numbers. We must calculate the amplitude points and how their position in the array relates to frequency based on what we know about the samples we took.

Basic FFT algorithms return an array of real values and an array of imaginary values. These represent the cosine and sine components of a set of phasors on the complex plane as rectangular coordinates. We use the equations from chapter 5 as given in equation (5.34) on page 158 to calculate the magnitude and phase of each point:

NOTES:

$$mag = \sqrt{x^2 + y^2} \quad phase = \tan\left(\frac{y}{x}\right) \quad (7.5)$$

where x is the real term and y is the imaginary term at each array index. With a simple "for" loop we can calculate the magnitude of each frequency value and store it in an array. The FFT algorithm returns duplicate (mirrored) information in the lower and upper halves of both arrays, so the loop and the analyses only use $N / 2$ values, which relate to the $N / 2$ frequency bins in the spectrum we ultimately create.

Use a "for" loop to calculate magnitude of each frequency point:

```
for (Index = 0; Index < N / 2; Index++)
    Mag[Index] = sqrt(Real[Index]^2 + Imag[Index]^2);
```

Magnitude is usually expressed in dB relative to the fundamental, so this calculation can be modified to:

```
dBMag[Index] = 20 * log10(sqrt(Real[Index]^2 + Imag[Index]^2) / Vfund);
```

where the dB relative reference is the amplitude of the fundamental as in

$$dBmag = 20\log\left(\frac{Vmag[Index]}{Vfund}\right) \quad (7.6)$$

Fundamental correction

This brings us to the question "What is the amplitude of the fundamental?" Recall that the amplitude of the fundamental was reduced with a notch filter so the harmonics and noise could be amplified by a factor of 100. But it is clearly an important part of our calculations. Ultimately it is 0dB in a spectrum graph, but its true value is needed in the magnitude ratio calculation of dB level for all other harmonic and noise components of the digitized samples.

The value in the frequency bin for the fundamental is wrong and cannot be used as the reference voltage in this calculation. We know the value, however, because we measured the zero and full scale of the DAC under test as discussed early in this section. By subtracting the zero scale measurement from the full scale measurement we get the full scale amplitude range of the DUT output signal as a voltage magnitude = $2Vpeak$.

The peak value of full scale is used as the denominator reference level V_{fund} in the $dBmag$ calculation. Insert this value into array index [M], which is the x axis location for the fundamental. With all other values less than the fundamental, 0dB is the maximum amplitude level and all others are negative dB (because the log of a ratio less than zero is negative).

NOTES:



When calculating dB, remember that $\log(0)$ is undefined, so it may be necessary to add a test for a magnitude of 0 before calculating the log. With the fundamental value and 0 value corrections, the complete algorithm is

```
RefLevel = (Vfs - Vzs) / (2 * sqrt(2)); /* RMS value of full scale sine wave */
for (Index = 0; Index < N; Index++)
{
    if (Index == M)
        dBMag[Index] = 0;
    else
    {
        Magnitude = sqrt(Real[Index]^2 + Imag[Index]^2);
        if Magnitude = 0 then
            Magnitude = 1E-8;
        dBMag[Index] = 20 * log10(Magnitude / RefLevel);
    }
}
```

Because this is a ratio algorithm and the ratio of the fundamental to itself is 1, or 0dB, the above algorithm simply puts 0dB into bin M . Note that the fundamental level can be corrected by adding the dB value of notch filter attenuation to the dB amplitude value in the fundamental bin M .

When $Magnitude = 0$, this algorithm replaces it with a magnitude of 10^{-8} , which calculates to -160dB. Software on mixed signal test systems often has a *magnitude* routine which will automatically (and quickly) perform these calculations; use it if it is available.

Note that it may not be possible to have a notch filter that only affects the fundamental. In the case where the notch filter attenuates amplitude values in other frequency bins, it is necessary to characterize the filter path and store correction factors for each bin in the measured bandwidth. These correction factors are then used to compensate the frequency spectrum from the sampled signal.

There now exists in capture memory a set of $N / 2$ magnitude points which represent the amplitude in dB of each frequency bin in the calculated spectrum. This array is all that is necessary to calculate SINAD, SNR and THD.

NOTES:

Creating a Spectrum Graph

We have the points for the vertical axis of a frequency spectrum plot. To plot amplitude versus frequency, we must calculate the horizontal points on the frequency plot which match each vertical point, allowing a magnitude vs. frequency graph to be plotted.

Frequency Points

Recall that there are N samples which were taken at frequency F_s and that the maximum frequency information contained in the FFT data is the Nyquist frequency $F_s / 2$. Also, recall that there are $N / 2$ amplitude points where each point represents a frequency bin. Using this information, we have an x (frequency) axis which ranges from 0 to $F_s / 2$ and a set of $N / 2$ points, with each point representing a frequency increment of $(F_s / 2) / (N / 2) = F_s / N$. The F_s / N quantity is the *Fourier Frequency* or *Fres* as discussed in *UTP, Fourier frequency, frequency bins and resolution* on page 185.

In simple terms, the frequency axis goes from 0 to $N / 2$ bins, with each bin representing an increase in frequency of (F_s / N) Hz. Knowing that the WD used $F_s = 204.8\text{KHz}$ and $N = 1024$, each frequency bin represents 200Hz and the frequency axis spans $200\text{Hz} * 512 = 102400\text{Hz}$. The fundamental will be in bin $M = 5$ at 1000Hz.

NOTES:

Synchronization Issues

Did you notice that, in the dynamic tests, there are 2 sets of samples being created? The DAC creates analog sample points and the waveform digitizer creates digital sample points. A portion of 7.2 dealt with synchronizing the WD sampling with the F_t of the DAC output. In our discussion, we assumed that the clock which feeds the digital vectors (sinusoid data points) to the DAC under test runs at F_s and that the waveform digitizer runs at its own (unrelated) F_s . This can only occur with 2 independent clocks. What if the test system has only a single master clock to which both the digital vectors and the waveform digitizer are synchronized?

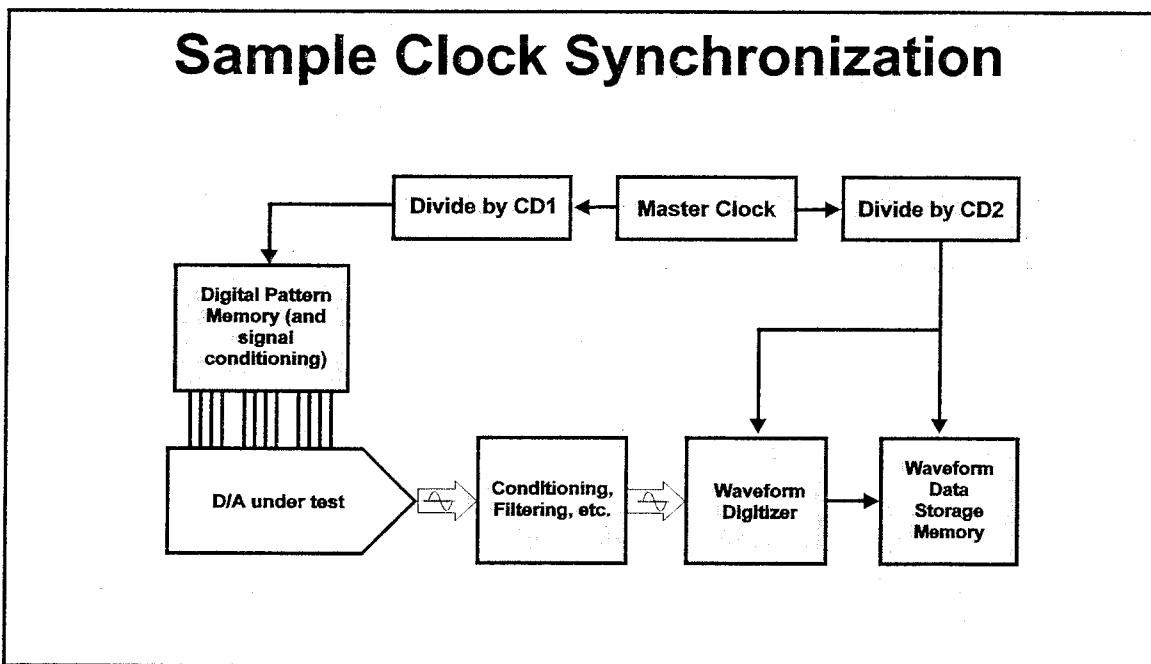


Figure 7.9

The clock division factors (CD1 and CD2) in the figure above must be integers if a binary divider is used. In this situation, the task of calculating the DUT F_s and the waveform digitizer F_s just got harder! Now the two F_s values must also be related to the master clock by integer factors. For this scenario, note that these relations exist between the DAC under test and the waveform digitizer:

- ◊ F_t is the same for both
- ◊ $\text{Digitizer } F_s = \text{MasterClock} / \text{CD2}$
- ◊ $\text{DUT } F_s = \text{MasterClock} / \text{CD1}$

NOTES:

Digital to Analog Converter Dynamic Parameters

N / M can be selected independently for digitizer and DUT but must conform to the relations above and the coherency equation.

If phase locked loops can be used, the clock division factors can be virtually anything. Phase locked loops suffer from a potentially long settling time (the time for the divided clock output to become stable) and from jitter. We know that jitter in the sample clock appears as amplitude distortion in a time sample set and as magnitude and phase distortion in its FFT generated spectrum. (See chapter 6, page 183).

Clock synchronization is a complex issue and calculation of DAC and waveform digitizer sample frequencies to achieve (something near) the desired test frequency from the DUT and the required sample set from the digitizer. It generally requires iterative calculations—

1. calculate DAC F_t based on F_t , M and N
2. adjust DAC F_t to meet test system clock frequency and resolution restrictions
3. recalculate DAC F_t with new DAC F_t (remember that F_t may not be exactly as stated on specification)
4. calculate a digitizer F_s , M and N based on the DAC's F_t (remember that F_t is the same for both DAC and waveform digitizer)
5. adjust digitizer F_s , M and/or N to meet test system clock frequency and resolution restrictions to achieve required F_t
6. if necessary, readjust DAC F_t , M and N to match F_t required by digitizer.

Ultimately the F_t may not be exactly as stated in the specification, but it should be as close as possible while still coherently generating and sampling the DAC waveform.

NOTES:



Key Points of This Chapter

- ❖ Testing of dynamic parameters is a complicated interaction of analog hardware, digital hardware and software.
- ❖ A single set of samples can be used for SINAD, SNR and THD.
- ❖ When using coherent sampling, it may or may not be possible to create the exact Ft required by the DAC specification.
- ❖ The digital input codes to create a sine wave can be created with an algorithm or an inverse FFT. Once they are created, they can be stored and used for any frequency by changing the Fs sample frequency at which they are sent to a DAC under test.
- ❖ The DAC output must be smoothed with a filter to remove steps which cause high frequency distortion.
- ❖ The DAC output may need to be level shifted to match the WD input range.
- ❖ The DAC output signal must have an anti-aliasing filter for the WD, but this function may be done by the DAC's smoothing filter.
- ❖ The DAC output may need a notch filter to attenuate the fundamental signal component so the noise and harmonics can be amplified for the test system WD. The amplification should be as much as possible without exceeding the WD input range.
- ❖ For coherent WD sampling, iterative calculations and trial and error are required to find a set of WD sample parameters that meet the restrictions of the WD and match the Ft frequency of the DAC output signal.
- ❖ Samples taken with the WD must be compensated for conditioning (filters, etc.) done to the DAC output signal.
- ❖ Frequency information for FFT/DFT processed samples is obtained from WD sample parameters.
- ❖ Synchronization of DAC input codes and WD samples may be necessary on some ATE systems.

NOTES:

Answers to chapter questions

Answer 7.1: 20th harmonic; the filter attenuation starts at 20×10^4 ; $20000 / 1000 = 20$.

Answer 7.2: a) Solving for V_{dist} yields a total distortion sum of $790 \mu\text{V}_{\text{RMS}}$; this is the sum of all harmonic distortion! b) With 10 harmonics, the average harmonic amplitude is about $79 \mu\text{V}_{\text{RMS}}$.

Answer 7.3: No. A low pass filter would attenuate the harmonics and noise in the frequency band we are measuring.

NOTES:



Answers to Lab Exercise 7.1

Answer 7.1.1: 1000Hz

Answer 7.1.2: a) 5000 b) No c) F_s

Answer 7.1.3: a) 0, 4095, 2048 b) the fact that it is a 12 bit DAC

Answer 7.1.4: No. The number of horizontal points (5000) and the number of vertical points (4096) are much greater than the horizontal and vertical resolution of the graph on the screen (approx. 500 x 300).

Answer 7.1.5: 4999

Answer 7.1.6: 200nsec

Answer 7.1.7: a) 0 b) it is the center of a $\pm 1V$ sine wave; 2048 is the center of a 0 to 4095 range for the DAC input.

Answer 7.1.8: a) 4095 b) yes c) yes d) changes in the Y sine value are smaller than a 12 bit LSB, so the bit value stays the same e) it is not exact; i.e. it has step distortion, also called quantization error.

Answer 7.1.9: a) 2048 b) No c) No d) the signal is moving faster at 0; i.e. the rate of change of amplitude with respect to time is fast enough to cross an LSB boundary for each sample point.

Answer 7.1.10: a) It changed to 100KHz b) slightly different (if any)

Answer 7.1.11: a) yes b) with only 50 steps, the amplitude changes a lot with each sample c) stepped

Answer 7.1.12: a) 50 b) Yes, but only near 0 and full scale input (*Index* = 12,13 and 37,38) c) 4091 and 4 d) No

Answer 7.1.13: a) F_t increased by a factor of 3 b) 3 cycles c) No, the vertical step size depends on the rate of change of the sine wave at a given point in time.

Answer 7.1.14: a) 2.9MHz b) No c) No; For N=5000, a) 29KHz b) Yes c) Yes

Answer 7.1.15: a) no effect b) 31 c) Looks OK d) Vertical resolution is very low with a 5 bit DAC

NOTES:

Answers to Lab Exercise 7.2

Answer 7.2.1: $F_s = 341.3333333334e3$ Hz

Answer 7.2.2: a) No, the value does not lie on a 10Hz boundary. b) 10 Hz c) 341 330 Hz

Answer 7.2.3: a) 999.990234375 b) Yes c) No, there will be leakage. The expected F_t is 999.990234375Hz and the actual frequency being sampled is 1000Hz exactly. Thus the WD sample set will contain a slight amount of data from a second test signal period, causing the leakage.

Answer 7.2.4: a) 1 024 000 b) No, it is greater than the maximum WD sampling frequency of 1MHz.

Answer 7.2.5: Three possible choices are:

$N = 4096, M = 5, F_s = 819200, UTP = 5\text{msec}, F_{res} = 200\text{Hz}$

$N = 1024, M = 5, F_s = 204800, UTP = 5\text{msec}, F_{res} = 200\text{Hz}$

$N = 512, M = 1, F_s = 512000, UTP = 1\text{msec}, F_{res} = 1000\text{Hz}$

$N = 1024, M = 25, F_s = 40960, UTP = 25\text{msec}, F_{res} = 40\text{Hz}$

NOTES:

References

1. Mark Landry, "Production Testing of PCM (Digital) Audio Circuits", 1983 *Proceedings of the International Test Conference*, Institute of Electrical and Electronic Engineers, Inc. 1983, pp. 767-770
5. Howard M. Tremaine, D.Sc., FAES, *Audio Encyclopedia*, Second Edition, Howard W. Sams & Co., Indianapolis, IN 46268, 1973, pg. 1393.
6. Richard G. Lyons *Understanding Digital Signal Processing*, Addison Wesley Longman Inc., 1997, pp. 15-18
- . Ref. 1

NOTES:

Digital to Analog Converter Dynamic Parameters

NOTES:

Analog to Digital Converter Dynamic Parameters

Goals

- ❖ Know what dynamic ADC specifications represent
- ❖ Understand why conditioning and filtering of ADC input is necessary
- ❖ Understand how to use the ADC under test as a waveform sampler
- ❖ Understand the relationship between the number of bits of an ADC and the best case SNR (ENOB)
- ❖ Understand the principles of normal sampling and undersampling

Objectives

- ❖ Learn to calculate requirements for ADC input signal purity
- ❖ Be able to use a tester waveform generator to create an input signal which is adequate and sufficient for a particular ADC under test
- ❖ Be able to solve real world problems related to the test system, the characteristics of the ADC under test, unexpected results, etc.
- ❖ Be able to use tester or external conditioning circuitry to properly condition analog input signals before digitizing
- ❖ Know why undersampling is such a useful sampling technique and when to apply it.
- ❖ Use questions and interactive software laboratory exercises to demonstrate the often confusing concepts of coherent sampling, Fourier analysis and dynamic parameter calculation
- ❖ Bring to light the various options and trade-offs possible to allow an intelligent choice of a particular test technique for a particular device

What you will learn

- ❖ How to create a sinusoidal input signal sufficient to test a particular ADC based on its specifications.
- ❖ That ADC testing requires adjustments to the test system for each tested device, and what the adjustments are.
- ❖ How to use a filter to make an ADC input signal pure and to provide anti-aliasing.
- ❖ The benefits of using a track and hold on an ADC input and the problems of not using one.
- ❖ ADC input buffer circuits to prevent dynamic impedance problems.
- ❖ How to take a coherent sample set with the ADC under test.
- ❖ How to take a coherent sample set using undersampling.
- ❖ The origin of *effective number of bits* (ENOB).

NOTES:

ADC Dynamic Specifications

An analog to digital converter is a circuit that converts an analog signal into a sequence of digital numbers; the digital values represent points on the analog signal at the instant the conversions are made. The most fundamental ADC is a comparator—it is a 1 bit ADC which tells whether the analog signal is above or below a given reference value. By making the reference value 1 LSB apart for additional comparators, a *flash converter* is created, giving a binary output value which has all ones for the point above the signal and zeros for points below it. These outputs are then encoded to give a straight or 2's complement binary output (see Figure 8.1).

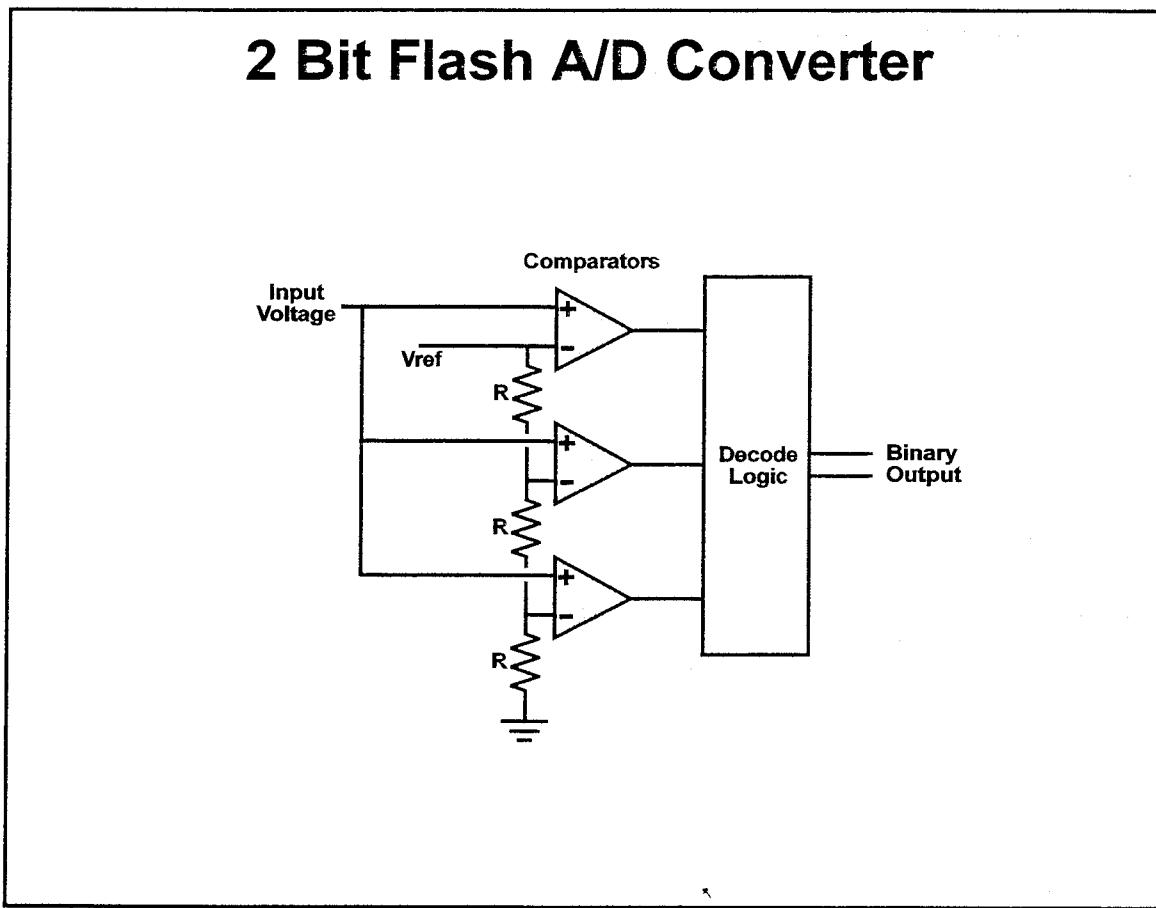


Figure 8.1

NOTES:

Analog to Digital Converter Dynamic Parameters

Dynamic specifications for an ADC describe how well the ADC can capture a dynamic signal of a specific frequency or perhaps containing multiple known frequency components. In the example of a flash converter, a sinusoid may be captured with the ADC under test, with the sine wave having a frequency at or near the Nyquist rate of the DUT. The degree of accuracy of the captured points as digital data determines the degree to which the device meets its dynamic specifications. The parameters we will discuss are

- ◊ Total harmonic distortion
- ◊ Signal to noise ratio
- ◊ Signal to noise and distortion ratio
- ◊ Intermodulation distortion
- ◊ Dynamic range and spurious free dynamic range

Signal to noise ratio (SNR)

SNR is conceptually the same parameter as that for an op amp, and, like THD, is analyzed by presenting a pure sine wave as the analog input signal and digitizing it with the ADC. A DSP algorithm does the work of extracting SNR information. Specified in dB.

Total harmonic distortion (THD)

While conceptually the same parameter as THD for an op amp, measurement of THD is different for an ADC. The input is a pure sine wave. The output is a set of binary values digitized from the sine wave which must be compared with the characteristics of an ideal sine wave. A DSP algorithm does the work of extracting THD information. Specified in dB.

Signal to noise and distortion ratio (SNDR or SINAD)

SNDR is the ratio of the fundamental to the combined noise and distortion. Specified in dB.

Intermodulation Distortion (IM)

IM is a test for non-harmonic product terms that appear in a device signal due to undesired modulation of two frequency components of a signal. This modulation is a result of non-linear characteristics within the device under test. The test is performed by putting a summed two sinusoid tone into a device and looking for frequency components in the sum and difference frequency bins of the two sine frequencies.

Dynamic Range and Spurious Free Dynamic Range (SFDR)

Dynamic range is defined as the ratio (usually in dB) of the maximum signal size to the minimum signal size; for an ideal ADC it is $20\log(2^{\text{bits}} - 1)$. Spurious Free Dynamic Range is the ratio of the primary signal (carrier or fundamental) to the largest spur (any frequency peak that is not the fundamental), which could be from distortion or harmonics.

NOTES:

Testing ADC Dynamic Parameters

Before discussing how to test dynamic ADC parameters, let's review these things:

- ◊ An ADC data sheet
- ◊ A general test system configuration for ADC dynamic parameter testing
- ◊ A specification
- ◊ A diagram of the errors
- ◊ A list of ADC dynamic parameters and the measurements required to calculate their values

NOTES:

Test System Configuration for ADC Dynamic Parameter Tests

As you can see in Figure 8.2, the WD and DSP components of a mixed signal test system are key elements in testing dynamic parameters. The basic approach to testing SNR, THD, etc. is to create as pure a sinusoidal wave as possible, then analyze it to see how pure it really is. Requirements on the digital subsection of a test system are more stringent for testing dynamic parameters versus static. Coherent sampling requires synchronized digital and analog subsystems.

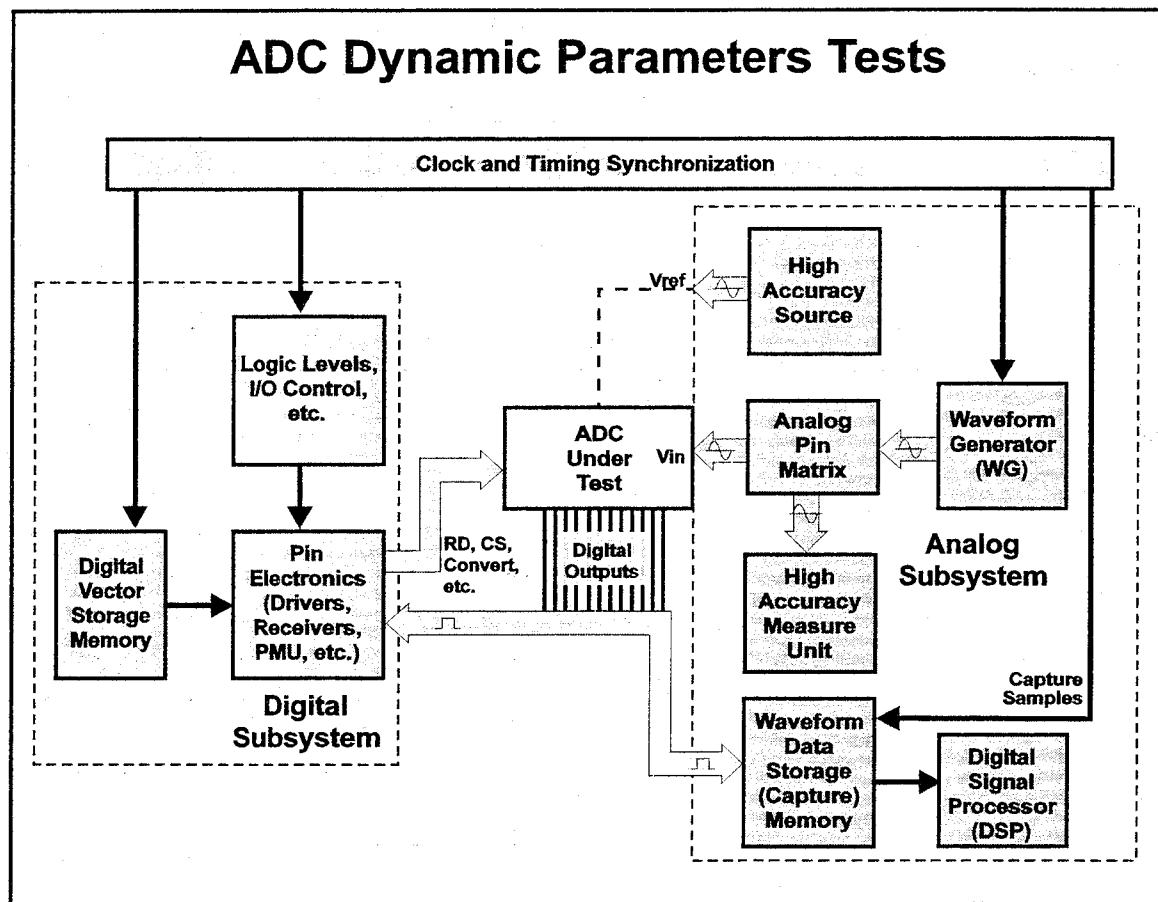


Figure 8.2

NOTES:



Example ADC Data Sheet

Now it is time to examine ADC specifications, see which ones are considered "dynamic" and learn how to measure them on a mixed signal ATE system. The following table is an example of an ADC data sheet specification; it has dynamic parameters similar to a DAC—SNR, THD, SINAD, IM and input range. Note that the AC specifications are discussed in the section *Other Important Parameters* on page 95. Whether the digitized data from a waveform is from a test system WD or from an ADC under test, the calculation of dynamic parameters is the same. Thus the discussions about calculating SNR, THD, etc. starting on page 250 are just as valid for ADC testing as for DAC testing.

Parameter	Conditions	Min	Typ	Max	Units
OUTPUT					
Resolution		14			Bits
Output format	straight binary				
STATIC					
Offset error	Output code = 0x0			0.8	% FSR
Gain error	Output code = 0x3FFF			2.0	% FSR
Differential nonlinearity	no missing codes to 14 bits			± 1	LSB
Integral nonlinearity				$\pm \frac{1}{2}$	LSB
Input range		0 to +4V			V
DYNAMIC					
SNR	$f_{in} = 1000\text{Hz}$ sinusoid	76	80		dB
THD	$f_{in} = 1000\text{Hz}$ sinusoid		-78	-70	dB
IM	$f_{in} = 1000\text{Hz} + 7000\text{Hz}$ tone			-72	dB
AC					
$t_{acquisition}$	After Busy signal goes inactive		200	500	nsec
$t_{aperture}$	After Start Convert signal goes active		10	20	nsec
Conversion time				25	μsec

Table 8.1 Example ADC Specification

NOTES:

Parameter Measurement Requirements

Measuring dynamic parameters requires coordination of a number of test system and DUT elements. Fortunately, once everything is set up properly, one set of sample points will provide the data for several parameter calculations. Following is a table showing the measurement requirements for SINAD, SNR, THD and IM. SINAD is not in the specifications in Table 8.1 but is included because understanding it helps to understand SNR and THD.

Parameter to test	Measurement(s) required	Items needed for measurement(s)
SINAD, SNR, THD	1. Zero scale 2. Full scale 3. A set of sample points of ADC output 4. DSP frequency analysis	◆ Analog sine wave input ◆ F_t ◆ F_s ◆ M ◆ N
IM	1. Zero scale 2. Full scale 3. A set of sample points of ADC output 4. DSP frequency analysis	◆ Tone that is sum of 2 sine waves for analog input ◆ F_t ◆ F_s ◆ M ◆ N

Table 8.2 Measurement requirements for dynamic parameter testing

NOTES:



Items and Steps Required to Dynamically Test an ADC

Many different software, algorithmic and hardware items are needed to do a dynamic test on an ADC, but fortunately not as many as a DAC. The following items will be discussed in this chapter:

1. a coherent sine wave must be generated by the WD as DUT input
2. a smoothing filter may be needed to create an input sine wave that is pure enough.
3. an anti-alias filter may be required at the ADC input (may be same filter as smoothing filter)
4. the ADC input signal must cover as much of the DUT full scale range as possible
5. a level shifter may be needed so the input signal fits within the ADC input range

When everything is properly set to test the ADC, the steps required to perform dynamic tests are as follows:

1. Make a continuous input signal with the tester for the ADC to convert.
2. Coherently collect a set of samples with the ADC.
3. Send the collected set of time samples to the DSP to perform DFT/FFT analysis
4. If necessary, convert rectangular frequency results to polar (magnitude and phase) results
5. Analyze the frequency bins of interest using equations or tester algorithms for SINAD, SNR, THD and compare to specification.
6. Make a pass/fail decision based on the results.

NOTES:

Creating an input signal

Dynamic parameter measurement is similar to testing a DAC in that a sinusoidal signal is captured. The difference is that, rather than using the test system digitizer to capture a sinusoidal waveform, we use the DUT to capture it. Also, the sinusoidal analog input waveform must be created by the test system. Our first task, then, is to create a "good enough" input signal.

There are several requirements which must be met when creating the input sinusoid for an ADC test:

- ❖ the input signal amplitude must be as large as possible, preferably extending from zero scale to full scale
- ❖ the input signal DC level must be offset by the same amount as the DUT offset
- ❖ the input signal maximum amplitude must not exceed either zero or full scale
- ❖ the input signal must have noise and distortion levels well below the distortion to be measured from the DUT (at least 10 times less)
- ❖ the input signal frequency must be either coherently timed with the sample frequency or the time samples must have a window function applied prior to performing an FFT or DFT

Adjusting for zero and full scale

Maximum input signal amplitude is required for the SINAD, SNR, THD and IM tests, otherwise the test does not represent the best DUT performance. The input sinusoid must have the maximum amplitude possible without exceeding the zero and full scale input values for the DUT. Zero and full scale measurements for an ADC are not trivial; the subject is discussed at length in chapter 4.

To maximize the signal to noise ratio, we must do 2 things:

1. maximize the signal
2. minimize the noise

When the input signal goes outside the zero or full scale values of the ADC's input, the output codes remain at their minimum or maximum and are no longer related to the input signal. Even if the input signal's amplitude is less than the DUT's full scale range, the DC offset of the signal must be matched to that of the DUT to keep the sinusoid in the linear range of the ADC under test. When this problem occurs it is called *clipping* the input signal.

Figure 8.3 on page 269 illustrates the situation with 2 sine waves. One has an amplitude that is too large and exceeds the ADC's minimum and maximum input levels. One has a smaller amplitude but has a maximum signal point which still exceeds the ADC input maximum level because of the input's DC offset. You can see that careful setting of the input signal is important *and is different for every unit tested!*

NOTES:

ADC Input Signal Clipping

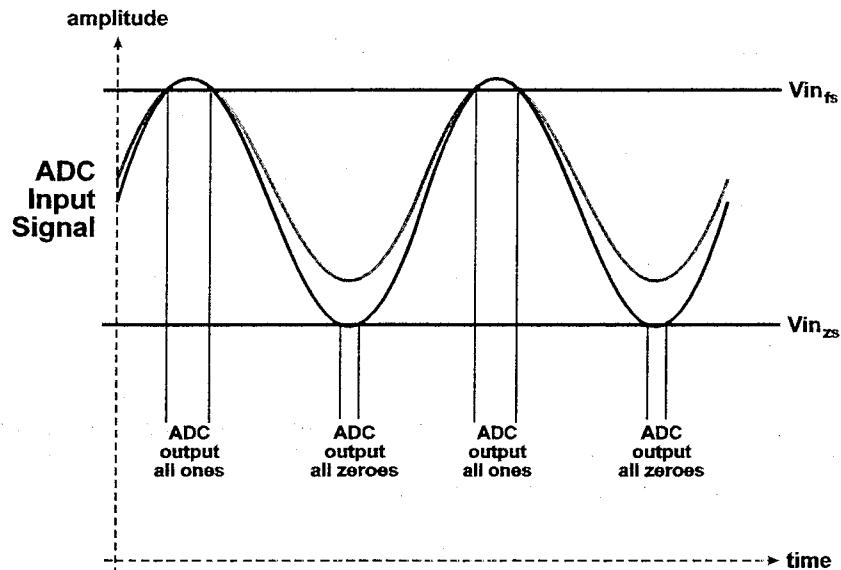


Figure 8.3

Question 8.1: Why is the input signal different for different units of the same device type?

Input signal purity filtering

In general, creating a clean sinusoid requires either a *very* good signal generator or a high Q analog low pass or band pass filter to remove everything from the signal except the pure sine wave. To test the ADC for SNR,

NOTES:

THD, etc., the input signal must have less distortion than the DUT (the continuous struggle of "Does the measurement reflect the state of the DUT or the test system?" as discussed on page 309). How much less depends, as usual, on the DUT. Table 8.3 lists the *best possible* signal to noise ratio of an ADC as determined by its resolution. This is based on the *quantization noise* as discussed in chapter 6 on page 181.

Bits	SNR (dB)
8	50
10	62
12	74
14	86
16	98
18	110

Table 8.3

The DUT input sinusoid must be better than these figures, with a rule of thumb of *at least* 10dB (3 times) better.

To purify the input sinusoid, a low pass or band pass filter is normally used to remove any undesirable frequency and noise from the signal, as shown in Figure 8.4. Compare this diagram to the one for conditioning a DAC output for a digitizer as shown in Figure 7.7 on page 242.

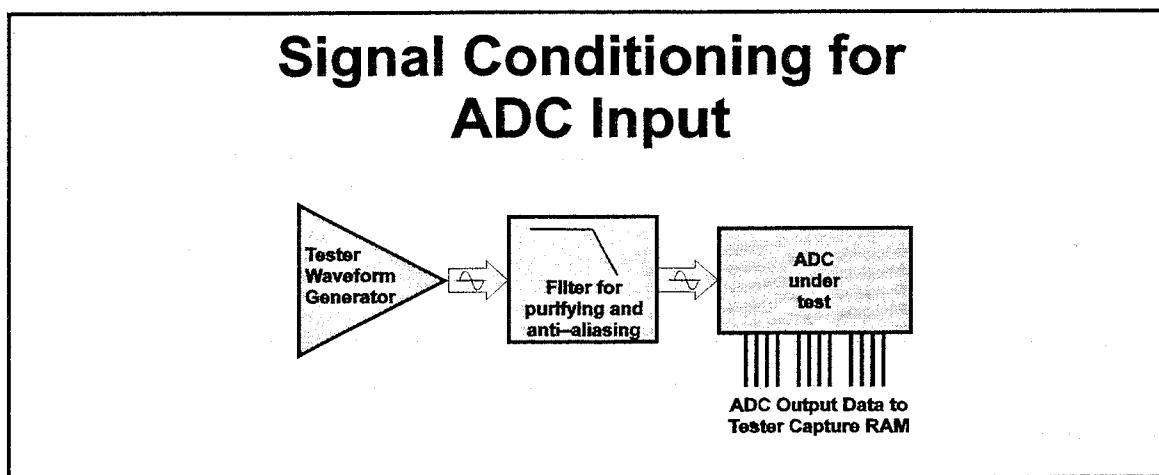


Figure 8.4

NOTES:



Input signal anti-aliasing

The Nyquist requirement for sampling requires a sample rate of more than two times the highest frequency component of interest. Viewed from the other direction, this can be stated that the input signal must contain no frequency components higher than $\frac{1}{2}$ the sample rate. To remove any frequency components above $\frac{1}{2}$ the sample frequency F_s , a low pass filter can be used. This is the *anti-aliasing* filter.

Fortunately the purifying filter also performs the anti-aliasing function and often the single filter is sufficient for both purposes. This is indicated in Figure 8.4, which shows a single block for the purifying and anti-aliasing filter.

As discussed in chapter 2 on page 52, a filter requires time for its output to settle after its input changes. To prevent a test program from waiting for the filter to settle for each DUT, often the tester waveform generator is set to run continuously and remains connected to the purifying filter. The DUT input signal is connected and disconnected with a relay or analog switch while a new DUT is inserted. If input signals of different frequency are required, the fastest way to test is to have several waveform generators and several different filters. Use relays or high quality analog switches to connect different signals to the ADC under test. Use a low impedance buffer to drive the ADC input.

NOTES:

Capturing the digital output data

Discussion of the digital interface is covered in the section *The Digital Side of ADC Testing*, on page 100. The figure below illustrates the capture of the ADC's digital output data. The method for capturing output data from a specific device type depends heavily on the output data format of the DUT. The important concept to understand regarding output data is that we must store a set of digital codes which are numbers representing samples of the analog input signal. These numbers are stored in the waveform storage memory, called also *Capture RAM*.

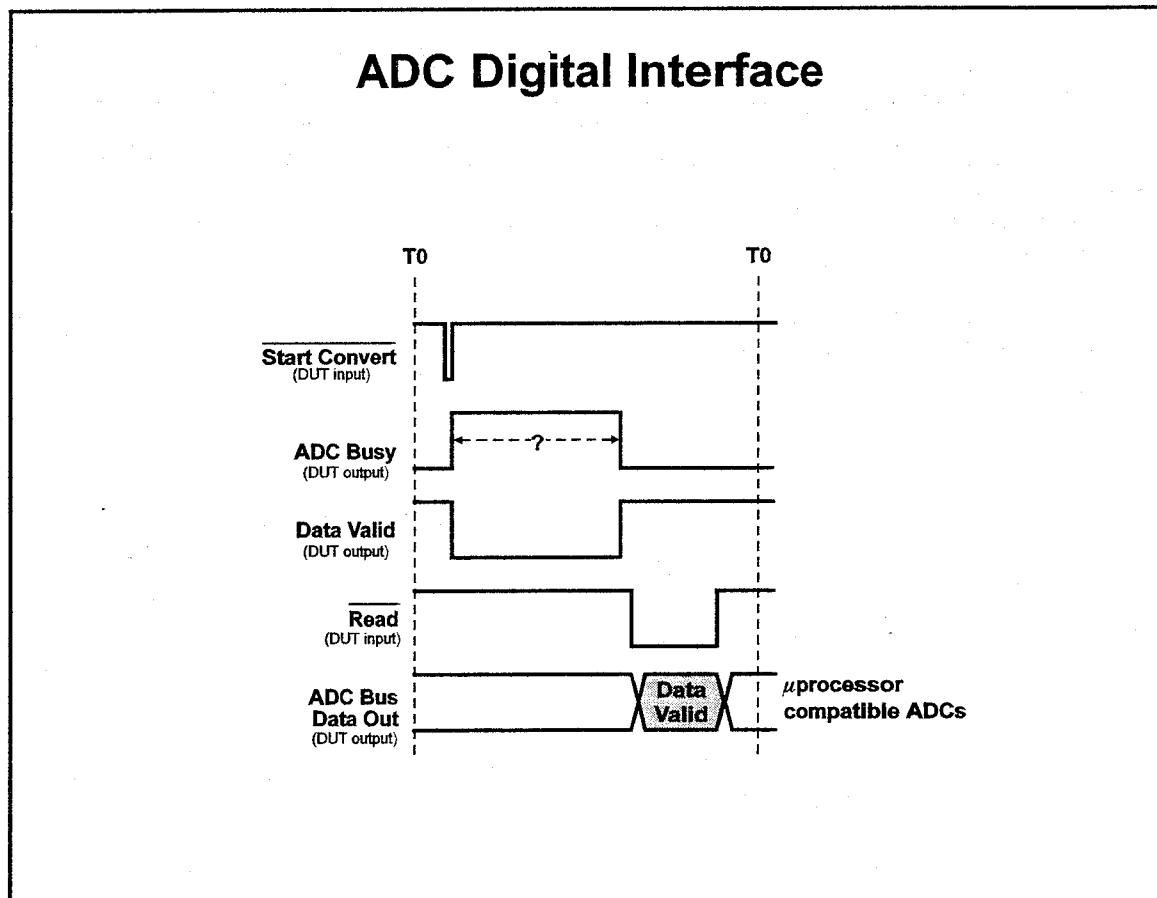


Figure 8.5

NOTES:

Acquiring and holding the input signal

An ADC requires that its input signal be constant while it is calculating the digital output code which represents that input. This puts some interesting constraints on the input, depending on the device being tested, and often requires a track and hold at the ADC input. To illustrate this, we will first consider an ADC without a track and hold.

An ADC with no track and hold

An example of an ADC without an internal track and hold is the AD574, which is a 12 bit device introduced by Analog Devices around 1980. Originally created with 2 chips, it had a 25 μ sec conversion time and no internal track/hold. Consider that, while the AD574 is converting, the input must not move more than 1LSB. If the input moves more than that, the conversion is actually being done on 2 or more different input values, causing the output result to be junk. This results in an *aperture time* of 25 μ sec. (See the definition of aperture time in chapter 4 on page 95.)

With a sinusoidal input, the question then becomes "Given the ADC aperture time, what is the maximum sine wave frequency that will yield a valid ADC output?"

First, we must know the maximum rate of change of a sine wave with respect to time. The equation for a sine wave is $y = V_{max} * \sin(\omega t)$. The maximum rate of change is given by the derivative with respect to time:

$$\frac{dy}{dt} = V_{max} \omega \cos(\omega t) \Big|_{max} \quad (8.1)$$

The cosine function is a maximum of 1 at $\omega t = 0, \pi, 2\pi, 3\pi$ etc., resulting in a maximum rate of change of the sine wave at those times. (Looking at the sine wave in Figure 8.6 on page 275, we can see where the maximum slope occurs by inspection.)

The equation for maximum slope thus becomes

$$\left(\frac{dy}{dt} \right)_{max} = V_{max} 2\pi f \quad (8.2)$$

Next, calculate an ideal LSB size, given FSR = 20V (for $V_{max} = 10V$):

NOTES:



We know the aperture time of the AD574 is the same as its conversion time of 25 μ sec, which imposes the condition that the input signal can change 1LSB per 25 μ sec, giving a rate of change of $2.44\text{mV} / 25\mu\text{sec} = 97.6\text{V/sec}$. What is the frequency of a $V_{\text{max}} = 10\text{V}$ sine wave which has that value as its maximum slope? Set equation (8.2) = 97.6 and solve for f , which yields

$$f = \frac{97.6}{2\pi \times 10} = 1.55\text{Hz}$$

Thus the maximum input signal frequency is 1.55Hz(!). Without a track/hold, the AD574 can only convert signals that change very slowly, such as a thermocouple.

NOTES:

Sine Wave Rate of Change

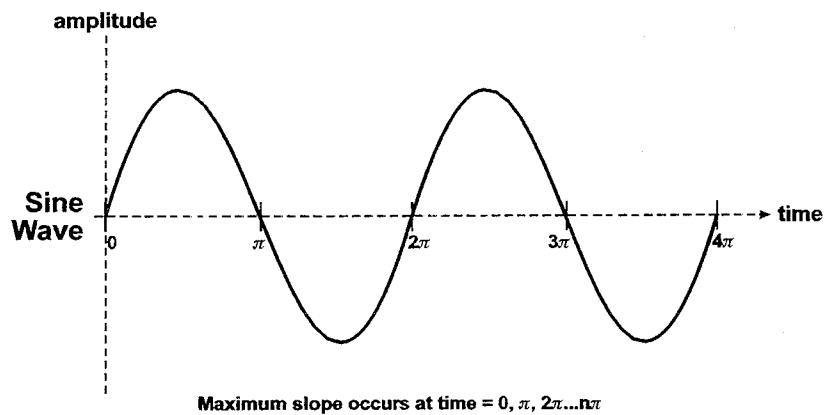


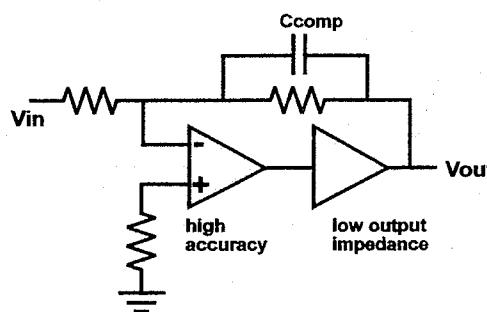
Figure 8.6

Adding a track and hold

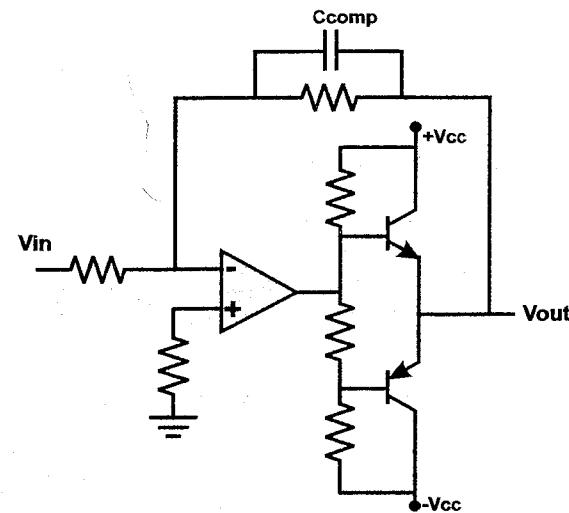
Putting a track/hold (T/H) in front of an ADC keeps the input signal constant while the conversion occurs. Thus the input frequency can be as high as the dynamic operation of the track/hold can handle. There are two important characteristics of T/Hs which must be considered when sampling a signal, although one of them can be ignored if the T/H is built into the ADC. They are *acquisition time* and *aperture time* (sometimes called *aperture delay*) and are illustrated in Figure 8.7.

NOTES:

High Speed Dynamic Buffers



Using Integrated Buffer



Using Discrete Buffer

Figure 8.8

The suggested buffers are high speed non-inverting unity gain devices which can supply a lot of current very quickly. The op amps are low noise high DC accuracy devices. The combination provides a low noise, low offset voltage and current, fast response driver. C_{comp} should be 5pF to 10pF to provide high frequency compensation (to prevent ringing due to stray capacitance at the input).

NOTES:

Sampling with the ADC under test

As discussed in chapter 6, the way to gather the most information about an analog waveform in the least amount of time is to use coherent sampling. To minimize the test time for an ADC, we use the familiar equation to set the relationship between the frequency F_t of the test (input) signal and the sample frequency F_s :

$$\frac{F_s}{N} = \frac{F_t}{M} \quad (8.3)$$

For the nominal value of F_t , recall the dynamic parameter specifications from Table 8.1 on page 265, duplicated here in Table 8.5 for convenience:

Parameter	Conditions	Min	Typ	Max	Units
SNR	$f_{in} = 1000\text{Hz}$ sinusoid	76	80		dB
THD	$f_{in} = 1000\text{Hz}$ sinusoid		-78	-70	dB
IM	$f_{in} = 1000\text{Hz} + 3100\text{Hz}$ tone			-72	dB

Table 8.5

The nominal value of F_t is noted under *Conditions* as $f_{in} = 1000\text{Hz}$. We calculated in Question 8.2 that the DUT can sample at a maximum rate of $1/(25.52\mu\text{sec}) = 39185\text{Hz}$, which is F_s for normal sampling (vs. *undersampling*, discussed in the section *Undersampling* on page 284). The fastest way to gather the samples we need is to get them all in 1 test signal period, i.e. $M = 1$. What do these values give us for N ? Substituting

$$N = \left(\frac{1}{\frac{25.52 \times 10^{-6}}{1000}} \right) \quad (8.4)$$

yields $N = 39.18495\dots$ which is not an integer nor is it a power of 2 (required for the fastest Fourier algorithm, the FFT). It also is not very many samples.

Rather than use $M = 1$, use the condition that $N = 1024$ and solve for M . The $F_s = 39185\text{Hz}$ is an upper limit for sample frequency; we can select any reasonable value slower than that. For $N = 1024$, a handy sample frequency is 20480, giving a nice round value of 20 for the ratio F_s / N .

Substituting into the coherency equation, with $F_t = 1000$ gives $M = 50$. Using an even value for M means we are getting 2 sets of essentially duplicate points over 25 cycles each, so it is just as valid to use $N = 512$ and $M = 25$, which cuts the time for sampling in half.

NOTES:

Analog to Digital Converter Dynamic Parameters

Question 8.3: a) What is the UTP (the time for sampling) for these parameters? b) What is the maximum harmonic that will be contained in the frequency data? c) What is the frequency resolution (FF or F_{res}) of the FFT spectral data?

NOTES:

Lab Exercise 8.1—Coherently sampling a sine wave

Note: if you worked Lab Exercise 6.2 and Lab Exercise 7.2, this one is very similar and may be skipped if you feel confident that you understand the concepts of sampling. However, it is fairly short and is a good review if you have the time.

Lab goals

- ❖ Create and view a continuous sine wave by making a Fourier series which has only 1 term
- ❖ Set sample parameters F_s , N , and M to create the F_t required for the input signal of the ADC under test.
- ❖ View the samples of the continuous sine wave
- ❖ View a spectrum created by a Fourier transform of the samples
- ❖ Use an FFT to do the time to frequency conversion
- ❖ Calculate various “secondary” parameters such as UTP

Lab objectives

- ❖ Review the process of digitizing samples of a continuous waveform
- ❖ Relate the sampling process to a DUT that is an ADC (rather than to a digitizer as in a prior lab)
- ❖ Examine the spectrum created by an FFT algorithm from the digitized samples

Initialize the lab, create a sine wave and sample it

Start the *DSP Lab* software. Press the *Spectrum Analysis* button and, in the *Fourier Waveform* window, set the following:

1. *Harmonic Number* = 1
2. *Peak Amplitude* = 1
3. *Term Type* = *Sine*
4. *Fundamental Frequency* = 1e3
5. *Overall Scale Factor* = 1

Go to the *Oscilloscope* tab and examine the waveform. Set *Show Samples* so it is **not** checked. Adjust the *Amplitude* and *Time Scale* knobs to see the wave clearly. Notice the time and amplitude displayed in the status bar at the bottom of the lab window when the mouse cursor moves over the graph.

Lab Question 8.1.1: What is the frequency?

NOTES:

Lab Question 8.1.2: What is the wave type (sine or cosine)?

Lab Question 8.1.3: What is the phase shift?

Lab Question 8.1.4: What is the amplitude?

Set the sampling parameters

Open the *Sampler* and set $F_s = 20480$, $M = 25$ and $N = 512$ as calculated just prior to the start of this lab on page 279.

Lab Question 8.1.5: What is the calculated value of F_f ?

View the spectrum

Close the *Sampler* with the *OK* button. Change to the *Spectrum Analyzer* window.

Lab Question 8.1.6: How many frequency peaks occur in the spectrum?

Adjust the *Frequency* knob to a maximum bin of 50.

Lab Question 8.1.7: What is the frequency value at the peak?

Return to the *Oscilloscope* window and put a check in the *Show Samples* checkbox. Notice the small red triangle sample points shown.

NOTES:

Lab Question 8.1.8: How many total sample points should there be?

Adjust the *Time Scale* knobs until you see all samples. 50msec is a good setting.

Lab Question 8.1.9: What is the time at the last sample?

Lab Question 8.1.10: Is this the Fourier Frequency, the sample period or the UTP?

Lab Question 8.1.11: What is the Fourier Frequency?

NOTES:

Undersampling

Undersampling allows an ADC input to be driven at its maximum input frequency to test the bandwidth of the analog input circuitry. Even though the samples are taken at a relatively slow frequency, the input of the ADC must be able to accurately follow the input signal in order to get a clean spectrum from the digital output data.

The word “undersampling” often causes confusion, mostly because there are (at least) 3 different meanings for it. The first meaning is simple—sampling at a frequency less than twice the bandwidth of interest. The second meaning is from communications theory and describes a way of demodulating a modulated high frequency carrier by sampling at a rate related to both the carrier and the modulation signal; this is more accurately called “bandpass sampling.” Neither of these definitions is important to our discussion.

The third meaning, the one used in automatic testing and in digitizing oscilloscopes, is to take one (or less) sample per test signal period, moving the sample point in time by an amount $1 / F_s'$ within the period, where F_s' is the effective sample frequency. This allows the sampling of a *periodic(!)* high frequency signal with an ADC having a much lower maximum sample frequency. This is understood more easily by looking at Figure 8.10 on page 286.

Given a test frequency F_t , the test signal period is $1 / F_t$ and the true sample period is $1 / F_s$. Assume an *effective sample rate* = F_s' which is the sample rate if all samples are considered taken during a single cycle. If the true sample period $1 / F_s$ is chosen as $k / F_t + 1 / F_s'$ as seen in Figure 8.9, the true sample period is then given by:

$$\frac{1}{F_s} = \frac{1}{F_s'} + \frac{k}{F_t} \quad (8.5)$$

where k is the number of test cycles skipped before each sample is taken (Figure 8.9 has $k=1$). Solving for F_s yields a true sample rate given by

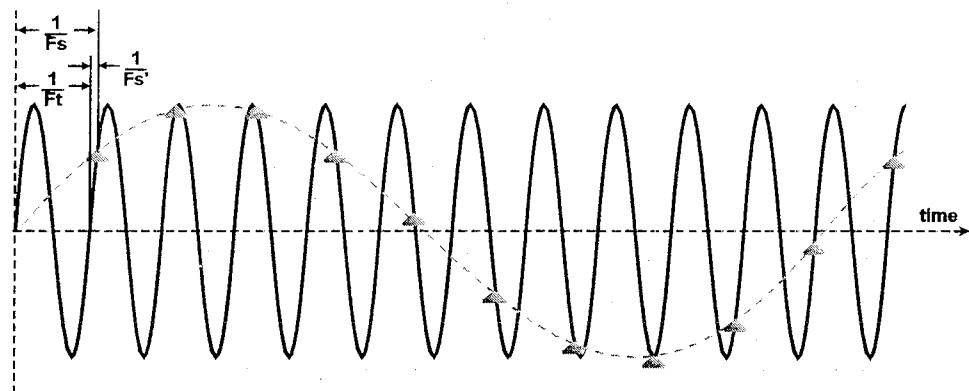
$$F_s = \frac{F_s' F_t}{F_t + k F_s'} \quad (8.6)$$

and for F_s' yields the effective sample rate given by

$$F_s' = \frac{F_s F_t}{F_t - k F_s} \quad (8.7)$$

NOTES:

Undersampling



▲ sample points, just less than 1 per test signal period

Figure 8.9

To determine the sample period for undersampling, follow these steps:

1. Choose a sample period based on the test signal frequency using the standard Equation (8.3) on page 279
2. If F_s is beyond the capabilities of the ADC, use equation (8.7) with $k = 1$ to calculate a new "undersample rate". If F_s is still beyond the capabilities of the ADC, pick a new (integer) value of k and recalculate.

What is k ? It is the number of test signal periods which occur per sample taken. If we are undersampling at one period per sample, $k = 1$; two periods per sample (or one sample per 2 periods) is $k = 2$, etc.

NOTES:

Recall that a set of samples of a signal contains no frequency information. Undersampling uses that fact to our advantage. Since the sampled waveform is periodic, each period looks exactly the same, so it does not matter from which period a sample is taken, as long as the data points are in the correct order to create a valid spectrum.

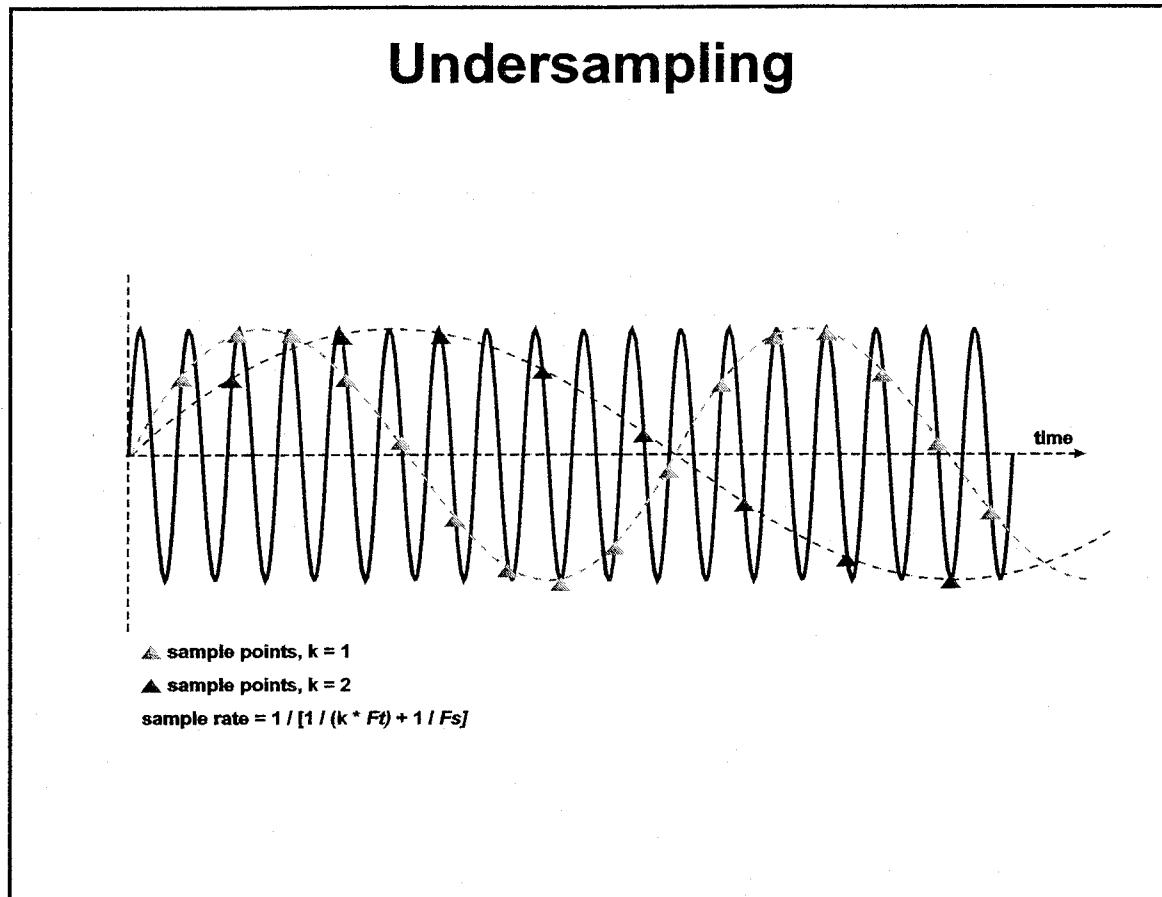


Figure 8.10

Other ways to perform undersampling are the “beat frequency” and “envelope” methods.¹ These are simple to set up for sampling, but the data points will be out of order for the envelope method.

NOTES:



The beat frequency method yields the same result as equation (8.7) with $k = 1$ but with an easier calculation...simply set $M = N + 1$. (To skip multiple cycles between samples, set $M = kN + 1$ with k = number of cycles to skip.) The envelope method sets $M = (N / 2) + 1$, taking a new sample every half cycle. The envelope method causes the samples to be “shuffled” in time and requires unshuffling prior to performing an FFT.

NOTES:



Lab Exercise 8.2—Undersampling with the beat frequency method

Lab goals

- ❖ Create and view a continuous sine wave by making a Fourier series which has only 1 term
- ❖ Set sample parameters F_s , N , and M to create the F_t required for the input signal of the ADC under test, with $M = N + 1$ for undersampling.
- ❖ View the samples of the continuous sine wave and note how they are taken at less than one per test signal cycle.
- ❖ View a spectrum created by a Fourier transform of the samples

Lab objectives

- ❖ Understand the technique of undersampling with the beat frequency method

In this lab you will see that undersampling is a valid technique for exercising an ADC at its worst case input frequency. It uses a method of setting $M = N + 1$ to take a sample at the rate given by equation (8.6) on page 284. Start the DSP Lab application, press the *Spectrum Analysis* button and, in the *Fourier Waveform* window:

1. press the *Clear Harmonics* button
2. set *Harmonic Number* to 1
3. set *Peak Amplitude* = 1.

Change to the *Oscilloscope* and press the *Sampler* button. Set it to *Calculate Ft*.

1. set $F_s = 512e3$
2. $N = 512$
3. $M = 513 (M = N + 1)$

Notice the calculated value of F_t . Is that a value that makes sense? (Yes.) Close the *Sampler* and, in the *Oscilloscope* set *Amplitude* = 2V and *Time Scale* = $2\mu\text{sec}$. You should see just over one cycle of the waveform.

Lab Question 8.2.1: Using $f = 1 / t$, **a)** what is the approximate frequency of the waveform? **b)** What frequency from the *Sampler* parameters does it (almost) match?

Lab Question 8.2.2: Make sure the *Show Samples* checkbox is checked. How many samples do you see?

Lab Question 8.2.3: Change the *Time Scale* to 2msec to put in enough cycles to see all samples. What is the wave shape of the red sample points?

NOTES:

Lab Question 8.2.4: Open the *Sampler*, change M to 1 and notice that the plot of samples looks the same, but the sampled wave is different.

What has occurred? A set of sample points has been taken from an input waveform with a frequency ($F_t = 513000\text{Hz}$) much higher than the maximum conversion rate of the ADC. The input frequency has exceeded the Nyquist rate of $F_{in} / 2$. Yet the points have no frequency information, they are amplitude points only. By knowing the input signal frequency, we can use these points to calculate the various test parameters such as SINAD or THD.

Lab Question 8.2.5: To prove that the data is real, in the *Oscilloscope*:

1. open the *Sampler*
2. set M back to 513
3. go to the *Fourier Waveform* window
4. set the *Add Noise* knob to 500.

Check the waveform in the *Oscilloscope*. Naturally the noise is not large enough to be visible there, but look in the *Spectrum Analysis* window (set *Frequency (maximum bin)* = 500) and you can see a visible noise floor—proof that the noise is affecting the sample points.

What is the *equivalent sampling frequency*? We know that $F_t = 513000\text{Hz}$ and we took enough samples for one full cycle, so use $M' = 1$. With these values and $N = 512$, calculate the equivalent sample frequency $F_s' = N * F_t / M' = 262,656,000\text{Hz}$. Lab Exercise 8.2 demonstrates that a high speed signal can be sampled with a “slow” ADC. This is only valid, however, if the input bandwidth of the ADC is higher than the maximum frequency in the input signal. In other words, the ADC input must have a track/hold that can track a 513KHz signal.

NOTES:



Lab Exercise 8.3—Undersampling with the Envelope method

The reason for the name “envelope method” of undersampling will become apparent with this lab. The method takes *almost* two samples per period by setting $M = N / 2 + 1$ (with N a multiple of 4).¹

Lab goals

- ❖ Create and view a continuous sine wave by making a Fourier series for a triangle wave.
- ❖ Set sample parameters F_s , N , and M to create the F_t required for the input signal of the ADC under test, with $M = N + 1$ for undersampling.
- ❖ View the samples of the continuous sine wave and note how they are taken at less than one per test signal cycle.
- ❖ Use an FFI to do the time to frequency conversion.
- ❖ View a spectrum created by a Fourier transform of the samples.

Lab objectives

- ❖ Understand the technique of undersampling with the envelope method

In this lab you will see that envelope undersampling is a way to take samples for an input signal frequency that runs just over the Nyquist rate. It sets $M = N / 2 + 1$. This exercise uses a triangle wave Fourier sum rather than a pure sinusoid to show that these techniques work on any waveform. Start the DSP Lab application, press the *Spectrum Analysis* button and, in the *Fourier Waveform* window:

1. press the *Triangle* button in the *Preset Waveforms* panel.
2. change to the *Oscilloscope*
3. press the *Sampler* button
4. set $F_s = 512e3$
5. set $N = 512$
6. set $M = 257$ ($= N / 2 + 1$).
7. change to the *Spectrum Analyzer*.
8. set the *Frequency (maximum bin)* = 500

Notice that the spectrum looks “backward.” Something is wrong. Change to the *Oscilloscope* tab and set *Amplitude* to 2V and *Time Scale* to 2msec. Make sure the *Samples* checkbox is checked. What do you see? (Hint: remember the name of this sample method?)

NOTES:

Setting $M = N / 2 + 1$ takes a sample of the ADC input wave just more than once every 1/2 signal cycle. If the beat frequency method samples at a rate slightly less than one per test signal period, the envelope method takes samples at an effective sample period slightly more than once per test signal *half* period. Thus if the input signal alternates above and below 0, the sign of each sample alternates from the prior one.

Change the *Time Scale* to $50\mu\text{sec}$ and you can see the sample points go from positive to negative. Samples taken this way and stored as taken are *shuffled* and not in the correct sequence to form what is called the "primitive wave." That is the name for a waveform which would be achieved if samples could be taken in a single cycle at the optimum sample frequency.

To allow a spectrum to be created with this data, its sample points must be placed in the correct time sequence before performing a DFT or FFT. Reference 1 by Mahoney has good information on unscrambling shuffled data points.

NOTES:

Calculating SINAD, THD, SNR and IM

All the hard work is done. Full scale and zero scale values have been measured. The offset and gain errors have been calculated. An input sine wave has been created at a frequency which is synchronous with the sample rate of the ADC under test. The DC offset and full scale amplitude of the analog input signal have been matched to the ADC zero and full scale. DUT output sample data have been captured and stored in an array. The only thing left are the calculations; for these, refer to the sections starting with *Creating a Spectrum Graph* on page 247.

NOTES:

Sine Histogram Testing

Histogram testing can be done with a ramp or a sine wave input. A ramp histogram test is a static test and is discussed in the section *Histogram Test for DNL and INL* on page 109. The sine histogram is a dynamic test because a sine histogram can be done at the maximum input frequency of the DUT whereas a ramp input normally changes only as fast as the conversion rate of the ADC under test.

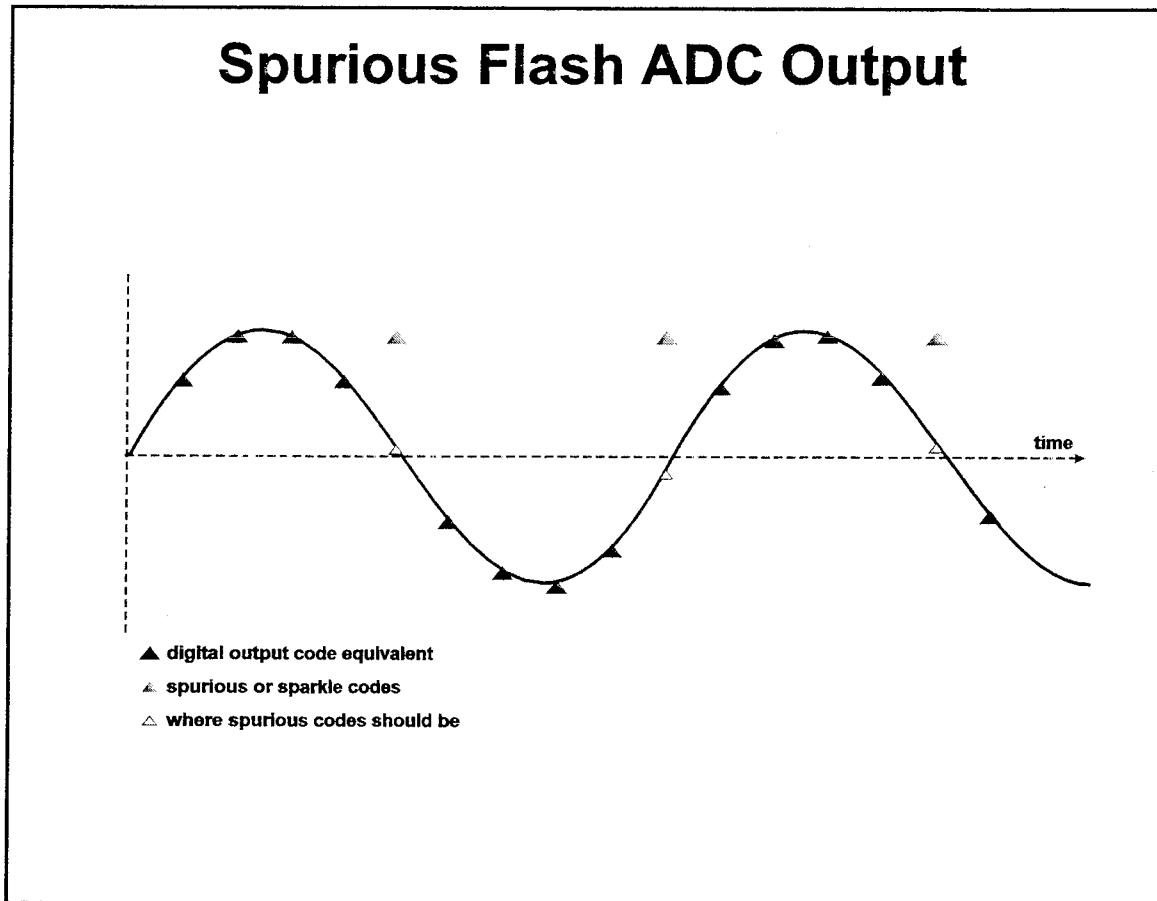


Figure 8.11

This test technique is most useful for testing flash ADCs. When the signal test frequency input to a flash ADC is raised to a point near the DUT input bandwidth limit or near the DUT's internal comparator response time,

NOTES:

the DUT exhibits a phenomenon called *sparkle codes* or *spurious codes*. This is when the output of the flash ADC goes to either full scale or zero scale because its internal circuitry could not properly decode the sampled input value into a digital output.

Notice in Figure 8.11 that, at the fastest moving portion of the input sine wave, the DUT output code goes to full scale rather than the expected point on the sine wave. Sine histogram testing can identify that spurious codes exist (ramp testing can show missing codes but not spurious codes).

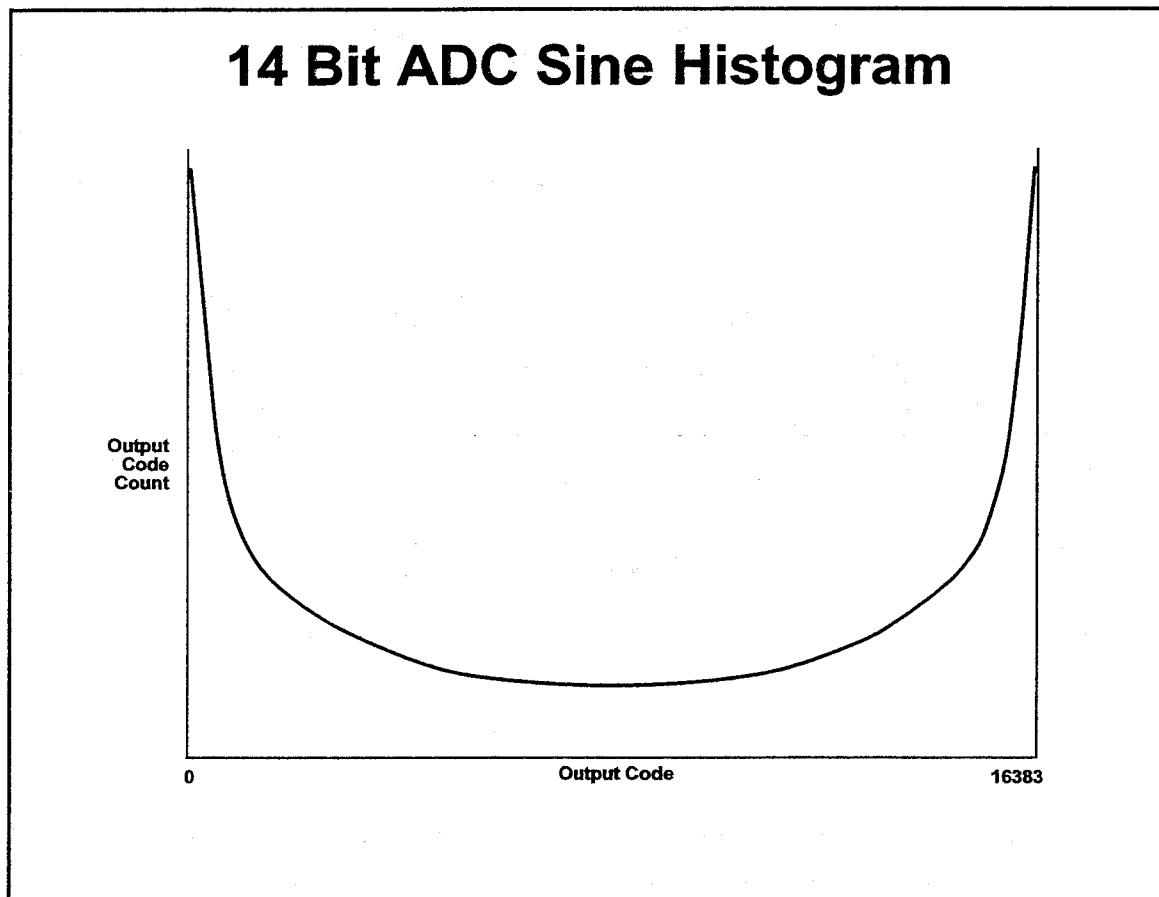


Figure 8.12

NOTES:

Sine histogram testing is more complicated than ramp histogram testing. Because of the shape of the distribution, analysis of the sine wave histogram is more complicated than that of the ramp. The code count for the fast moving portion of the input sine wave is less than the average code count, so there must be enough sample points taken to get a sufficient count of these codes.² This requires a longer test time to allow enough input cycles for all output codes to be sampled multiple times.

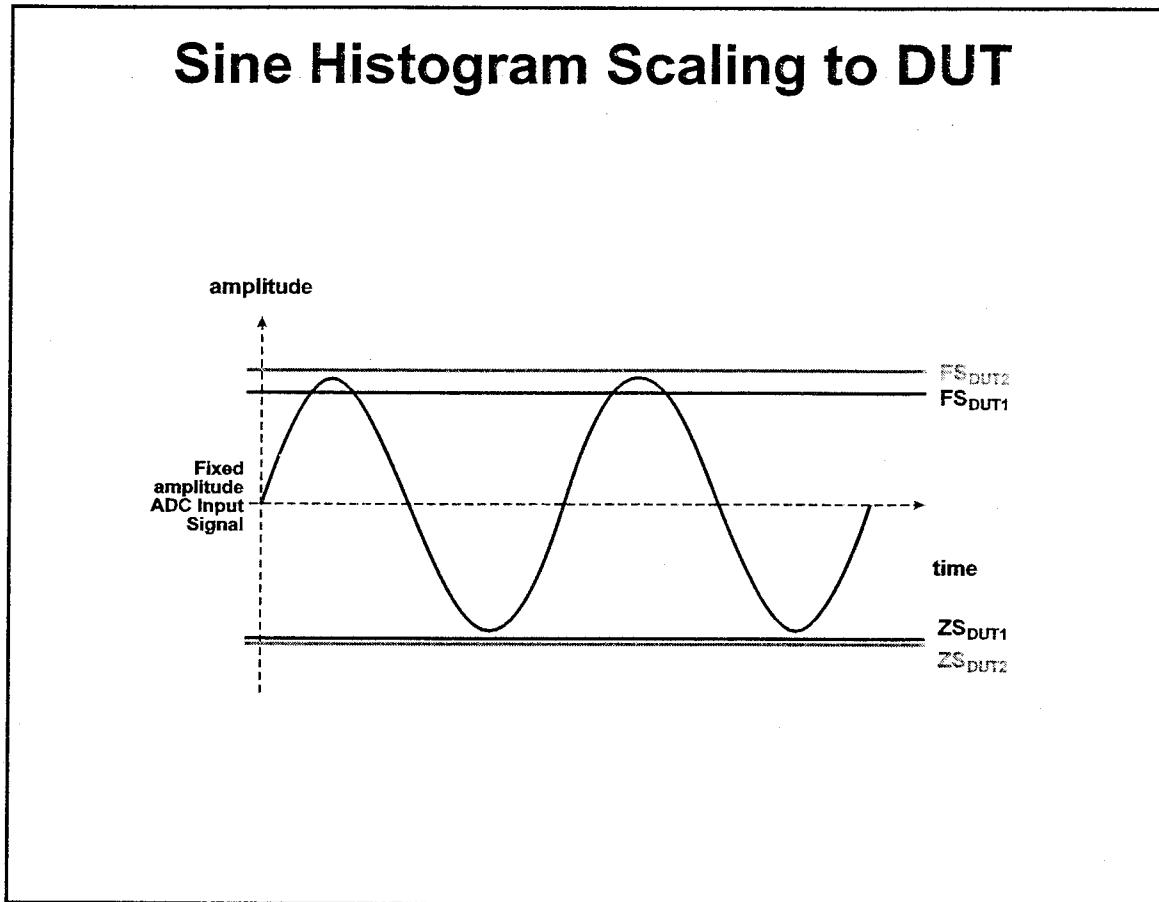


Figure 8.13

Statistically, the slow moving points on the sine wave have more counts and the fast moving points have less counts. The count distribution versus output code is shaped like a bowl as seen in Figure 8.12. The ADC input sine wave has a fixed offset and amplitude as set by the waveform generator, but each ADC under test has a

NOTES:

unique zero and full scale value. The DUT output codes are thus relative to the input characteristics of the DUT so the output code distribution must be separately calculated for each DUT. This can be done by using the DUT's full scale range and its offset in a software loop to calculate the points on an ideal sine wave. DNL error is then calculated by subtracting the ideal values from the actual points taken by the ADC.

This is a fairly involved subject; to learn more about it, get reference 1 listed at the end of the chapter. If you are interested in specifying spurious free dynamic range (SFDR), it may be more efficient to perform a distortion test at a specific high input frequency using undersampling. Then find the maximum distortion component and compare it to the fundamental. This does not identify at which codes spurious outputs occur, but it gives a quantifiable output performance parameter at a specific (presumably maximum) input signal frequency.

NOTES:



Effective Number Of Bits (ENOB)

A final INL representation to consider is that of *ENOB*. ENOB is an acronym for *Effective Number Of Bits*, which is a way of relating a signal to noise measurement (SNR) to a dynamic equivalent of integral linearity. ENOB is based on the inherent quantization noise of an ADC which, as discussed, is random. The final result of the ENOB/SNR relation is a very simple equation, but the origin of the constants in the equation is not at all obvious and the question of "Where did you get those numbers?" always arises.

The concept begins with the question of "Given that an ADC has known quantization error based on its resolution, what is the best SNR that an ADC can achieve?" (Here is where we delve more deeply into the world of statistics.) To model the quantization error, separate the ADC output into a perfect result and an error result as shown in Figure 8.14 on page 298. We use the following assumptions, which become more valid as the input signal becomes more "heuristic"³

- ❖ The error signal is random
- ❖ The error signal is uncorrelated with the perfect representation signal (this requires that, for a sine wave input, the sample set be the equivalent of a set taken from a single cycle, or from a single UTP, as the error will repeat in multiple sine wave cycles)
- ❖ The error is distributed statistically such that it is equivalent to "white noise"
- ❖ The error is distributed equally over the range of possible quantization errors

The last assumption is key to making the analysis easier—it assumes a uniform probability density of 1 for errors which range from $-\frac{1}{2}$ LSB to $+\frac{1}{2}$ LSB. Graphically, this is shown in Figure 8.15 on page 299, which indicates that the probability that an error will be between $-\frac{1}{2}$ LSB and $+\frac{1}{2}$ LSB is 100% (1) and is constant between those two points. Constant says it is just as likely that one value will occur as another. No errors can occur outside the $-\frac{1}{2}$ LSB to $+\frac{1}{2}$ LSB limits because that moves the error into the same range of a different ADC output code.

NOTES:

ADC Quantization Error

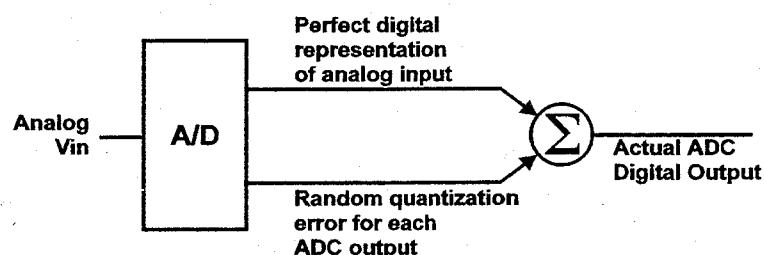


Figure 8.14

It can be seen that the average error is 0LSB, halfway between the 2 extremes. Just like the average value of a sine wave, we cannot get any useful information by averaging the noise. (We have already discussed that the errors can be averaged to zero.) However, we can use the same RMS concept used on a sine wave to get useful information about the noise. Square the noise, sum it then take the square root. The noise, in this case, is represented by the probability density function in Figure 8.15 and its RMS value is given by⁴ equation (8.8) for the uniform density distribution. This equation results from integration of the probability density function and is not derived here. Suffice it to say that the "12" is a constant of integration resulting from the area under the constant probability density curve shown in Figure 8.15 and is contained in the equation for variance of that distribution.⁵

NOTES:

$$\sqrt{\frac{(1\text{ LSB})^2}{12}} = \text{Noise}_{magnitude} \quad (8.8)$$

It should be pointed out that the quantity above is called the *standard deviation* in statistics, not the *RMS value*, but they are virtually the same thing. Defining 1 LSB in terms of the input signal peak value and the number of bits modifies equation (8.8) to:

$$\sqrt{\frac{\left(\frac{V_{max}}{2^{bits}}\right)^2}{12}} = \text{Noise}_{magnitude} \quad (8.9)$$

Quantization Error Probability Density

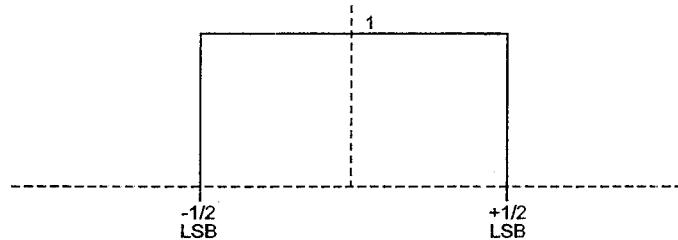


Figure 8.15

Because we are discussing RMS values, V_{max} for an AC waveform is 2 times V_{peak} , becoming (with some rearranging)

$$\sqrt{\frac{4 \times V_p^2}{12 \times 2^{bits} \times 2}} = \text{Noise}_{magnitude} \quad (8.10)$$

NOTES:

which can be simplified to

$$\frac{V_p}{\sqrt{3} \times 2^{\text{bits}}} = \text{Noise}_{\text{magnitude}} \quad (8.11)$$

Recall that we began this venture in an attempt to calculate the best possible SNR of an ADC. The noise portion of the ratio is given by equation (8.11). Now we need the magnitude for the maximum input signal to get the best signal to noise ratio. We must keep the units the same for signal and noise, so the maximum input signal is the RMS equivalent of Vmax, which, for a sine wave, is Vpeak divided by the square root of 2, to wit:

$$\frac{V_p}{\sqrt{2}} = \text{Signal}_{\text{magnitude}} \quad (8.12)$$

Dividing the signal by the noise yields

$$\frac{\frac{V_p}{\sqrt{2}}}{\frac{V_p}{\sqrt{3} \times 2^{\text{bits}}}} = \text{SNR} \quad (8.13)$$

which can be simplified to

$$\frac{\sqrt{3} \times 2^{\text{bits}}}{\sqrt{2}} = \text{SNR} \quad (8.14)$$

With some manipulation, this can be expressed in dB as

$$20(\log \sqrt{3} + \text{bits} \times \log 2 - \log \sqrt{2}) = \text{SNR}_{dB} \quad (8.15)$$

Calculating the constants gives the final result for SNR in dB of

NOTES:



$$6.02 \times bits + 1.76 = SNR_{dB} \quad (8.16)$$

Solve this equation for *bits* and you can get the equivalent number of bits which represent a device SNR figure—

$$ENOB = \frac{SNR_{dB} - 1.76}{6.02} \quad (8.17)$$

Now you know the origins of ENOB. Remember the conditions under which equation (8.17) is valid---

- ❖ The ADC must have a linear transfer function (e.g. not a companding or log device)
- ❖ The input signal's amplitude must span the ADC's full scale input range
- ❖ The input signal must be a sinusoid; otherwise the RMS value of input signal will be different from $V_p/\sqrt{2}$

NOTES:

Key Points of This Chapter

- ❖ Zero and full scale measurements are important because the DUT input signal must cover the ADC full scale range.
- ❖ The input signal must be more pure (have less noise and distortion) than the expected distortion to be measured from the ADC under test.
- ❖ The input signal may require DC offset to match the ADC input range.
- ❖ The ADC input requires an anti-aliasing filter.
- ❖ If the ADC has no internal track and hold, one will probably need to be used at the ADC input. Otherwise the analog input frequency may be required to be very low.
- ❖ Circuitry external to the ADC can disturb the analog input signal.
- ❖ The ADC input can disturb external circuitry that is driving the ADC input.
- ❖ Using the ADC in a mode of coherent sampling provides the fastest, most useful data
- ❖ Undersampling is a way to sample an input signal that is higher than the Nyquist frequency.
- ❖ Envelope undersampling requires the sample points to be *shuffled* to put them in correct order.
- ❖ Once the sample set is stored, SINAD, SNR, THD and IM are calculated as always.
- ❖ ENOB is a statistical method of relating SNR to INL.

NOTES:

Answers to chapter questions

Answer 8.1: Because the zero and full scale measurements of every DUT are different.

Answer 8.2: Maximum conversion (sample) rate is $1 / (\text{total conversion time}) =$

$$1 / (25\mu\text{sec} + 500\text{nsec} + 20\text{nsec}) = 1/25.52\mu\text{sec} = 39184.95\text{Hz}$$

Answer 8.3: a) 25msec **b)** the 10th harmonic; the maximum useful frequency information returned in the spectrum is $F_s / 2 = 10240\text{Hz}$; the 10th harmonic of 1000Hz is 10KHz, just inside the spectrum's maximum. **c)** 40Hz

NOTES:

Answers to Lab Exercise 8.1:

8.1.1: frequency = 1000Hz

8.1.2: it is a sine wave which starts at 0V when $t = 0$

8.1.3: no phase shift (0° or 0 radians)

8.1.4: amplitude is 1V peak (2V pk-pk)

(Note that this wave could also be considered a cosine wave with -90° phase shift. Verify this by changing the *Term Type* and *Phase Shift* in the *Fourier Waveform* tab if you wish.)

8.1.5: $Ft = 1000$ just as we calculated earlier

8.1.6: One frequency peak occurs because there is only one sine component in the waveform, with no noise and no distortion.

8.1.7: Frequency value at the peak is 1000.

8.1.8: There should be 512 total sample points.

8.1.9: 25msec

8.1.10: This is the UTP.

8.1.11: 40Hz.

NOTES:

Answers to Lab Exercise 8.2:

8.2.1: With the cursor, $t \approx 1.955\mu\text{sec}$ so $f \approx 512\text{KHz}$, although at the resolution of the graph, a small numerical error makes a big difference. It almost matches both F_s and F_t .

8.2.2: Two samples

8.2.3: A sine wave

NOTES:

References

1. Matthew Mahoney et al, *DSP-Based Testing of Analog and Mixed Signal Circuits Tutorial*, Computer Society Press of the IEEE, 1987.
2. *Ibid*, pg 147.
3. Alan V. Oppenheim and Ronald W. Schafer, *Digital Signal Processing*, Prentiss-Hall Inc., Englewood Cliffs, NJ, 1975, pg 415.
4. Richard G. Lyons, *Understanding Digital Signal Processing*, Addison Wesley Longman, Inc., 1997, pp 359–365.
5. *Ibid*, pp 481–484.

NOTES:

General Test Issues

Goals

- ❖ Understand some of the practical problems associated with mixed signal test program development
- ❖ Be aware of unexpected situations that can cause measurement errors
- ❖ Be aware of grounding and the ultimate objective of proper ground current routing
- ❖ Understand how DUT power supply impedance can affect DUT performance

Objectives

- ❖ Discuss the most basic problem associated with mixed signal measurements
- ❖ Present a clear way to think about ground current routing
- ❖ Present a rule of thumb for test system accuracy versus measurement limit requirements
- ❖ Discuss possible problems associated with DUT signal conditioning circuits
- ❖ Discuss power supply decoupling

What you will learn

- ❖ Ways to validate ATE measurements
- ❖ How good the measurement system must be compared to test parameter limits
- ❖ Ways to route ground currents for optimum measurement accuracy
- ❖ How DUT conditioning circuitry can affect measurement accuracy
- ❖ Strange things that can affect measurement results
- ❖ Items to check when a test program stops working
- ❖ DUT temperature can have a dramatic effect on measured results
- ❖ External circuitry on a DUT reference pin can affect gain and other parameters

NOTES:



Does the measurement reflect the state of the DUT or the test system?

This may be the most important question in mixed signal test development. It is the one you must answer when you develop a test program for a mixed signal device. The answer can be found only by comparing data-log results with independent measurements using another system, another test setup, a known device or no device at all.

The very nature of digital signals, with their fast transition times, tends to obscure information contained in analog signals. With a fast enough signal, any wire is an inductor and potentially a broadcast antenna. Any two conductors separated by a non-conductor (e.g. air or printed circuit material) create a capacitor. Digital edges are “fast signals”, so the potential for cross-talk between signal traces exists, especially between digital signal traces and high impedance analog nodes such as op amp or comparator inputs. There are situations in which mixed signal circuits must have a ground plane to shunt stray digital signals away from sensitive analog circuitry. Without a ground plane, these signals are coupled into the analog circuitry via parasitic inductive and capacitive elements. The task is to isolate the source of noise and prevent it from interfering with DUT measurements.

System noise

For example, consider a signal to noise measurement—is the measured noise truly the noise of the DUT or is it test system noise? How do you find out? If the test situation is one in which an analog signal is being digitized, you can remove the DUT, connect the analog signal pin directly to the local measurement ground (a.k.a. DUT zero) and run the SNR test. This will digitize the baseline noise for the test system with no DUT. Create a magnitude vs. frequency plot and compare it to one taken for the DUT in the same test setup. (Remember that averaging will give more repeatable results.)

System noise is a phantom, something that occurs but *should* not affect your measurements. If your DUT site is correctly designed and your timing is properly synchronized, system noise will not affect your test results. That is, however, much easier said than done. Fast digital signals cause transient currents through wiring traces, which capacitively couple into other wiring traces. System noise is generally not random, cannot be decreased by averaging (it may be *increased*) and can seriously degrade DUT measurements.

If the noise of the system alone is roughly the same as the noise of the system with DUT, then the results show that system noise is comparable to or higher than DUT noise. The system noise could come from any number of sources, e.g. uncorrelated digital noise due to poor grounding practice, digitizer noise due to improper signal filtering, RF or other interference, 60Hz noise, etc. The primary consideration in reducing system noise is paying attention to “where the currents will flow.”¹ This topic is discussed in more detail in *What is ground?* on page 312. Also, reference 1 at the end of this chapter is a great source of information on this topic.

To get an independent view of system performance at the DUT site, place the test program in a loop and look at the DUT site with a benchtop spectrum analyzer. Using a separate instrument to locate noisy node(s) can show noise problems due to, e.g., improper power supply by-passing or dynamic impedance problems.

NOTES:



DUT Noise

Noise exists in all physical systems. Although it can originate in many places, it is more evident with ADCs than other components.

DUT noise in ADCs

Noise in ADC testing is a fact of life. Although it has more effect on devices with higher resolution and smaller full scale range, noise occurs with all ADCs. Dealing with noise is an exercise in statistics and good load board design.

As discussed in the section on *Quantization Error* on page 181, all ADCs have an inherent error which can be characterized as random noise with a constant error probability over the width of an LSB. In other words, for a real world input signal which is more than just a sine wave, the quantization error is just as likely to be one percentage of an LSB as it is to be another.

The same is true for a sine wave, as the sample point value is dependent on the sample time as well as the sine wave amplitude, and the sample time depends on when sampling started, which is random with respect to the signal being sampled. (Recall that sampling an entire signal period is the basic requirement for coherent sampling; the first sample does not have to be at amplitude = 0 or any other special place. Mathematically, this means we can define time = 0 as any place we like.)

Bottom line: there is always *random* DUT quantization noise. With random noise, we know that the noise induced error is just as likely to be negative as positive, so averaging multiple data points for each transition can reduce the quantization noise to as low as you wish.

Question 9.1: What is the trade-off of averaging sample sets?

ADCs also use comparators, which must make decisions on whether a signal is above or below a particular level. Due to any or all the types of noise defined on page 47, the comparators will have random variations in their decisions when the input is very close to their threshold points. This is one of the primary causes of ADC transition noise as shown in Figure 4.6 on page 106.

Amplifiers Amplify Noise, Too

When dynamically testing DACs (or any DUT with an analog output) where you must remove the fundamental and amplify the harmonics and noise, be aware that any external noise which sneaks into the amplifier's input circuit will be amplified along with the signal. This can cause consistent or pseudo-random failures depending on the noise source. This can also occur at the input of active filters with gain greater than 1, summing circuits and any other place where signal amplification occurs.

NOTES:



Settling time errors

Another error source that appears to be random between different devices could actually be a situation where an analog signal is not allowed enough time to settle to its final value. If a signal is measured or digitized at less than its settling time limit, faster devices will be settled and slower devices will not. Don't make the mistake of using a "typical" settling time; if that is all that is specified, characterize the device for worst case settling and add some margin for error (e.g. double the worst case you found). Settling time errors can occur for DUT settling or for devices that drive the DUT (such as an active filter or ADC input buffer). Again a separate benchtop instrument may be necessary to locate the problem, although settling time is difficult to measure without very special care.

The test system must be 10 times better than the DUT

"Ten times better" is a general "rule of thumb" for measurements; if the measurement accuracy is only two times better than the signal specification limit, there is a 50% chance that a good device will fail because of the test system.

In making measurements with a digitizer, this rule becomes "the digitizer must have 4 more bits of resolution than the required measurement resolution. Four more bits represents a factor of $2^4 = 16$ times better; the next lower alternative is 3 bits, which is only 8 times better. Keep in mind that, with digitizers, the resolution is not the same as the accuracy. Relative measurements such as SNR may be OK with just a digitizer, but absolute measurements such as offset or gain may require a high accuracy meter rather than a digitizer.

NOTES:

What is ground?

The earth is the return path for current generated by power plants; moisture in the ground and the conductive minerals there, combined with essentially infinite cross-sectional area, provide a very low resistance return path for AC current. Thus the moniker "ground." This ground has minimal relation to the ground associated with measuring DUT parameters with an ATE system. (Ground is also a convenient way to make schematic diagrams easier to read.)

"DUT ground" or "DUT zero" represents the reference node for all DUT measurements. It must be stable and quiet. It can, however, be "floating" if necessary, as long as all other test system instruments and sources (including digital pin electronics) are referenced to DUT ground. There are many philosophies that claim to be "the best way to ground" a device or a system; as with most things, there are many ways to properly connect ground and even more ways to do it improperly.

The issue of primary concern in connecting and grounding a DUT site is "where do the currents flow?"¹ Some DUT circuits have a separate digital and analog ground pin and some do not. Some test systems have an active "driven" DUT ground; some have separate ground returns for analog and digital signal sources and some have a single common ground plane. Some ATE has optical isolation for digital signals. Whatever the situation, if you pay proper attention to where the currents flow, you can minimize or prevent grounding problems.

The general rule for current flow is that all currents should take the shortest return path back to their original source. This means you must know where each current originates; e.g. digital DUT input currents come from digital pin drivers and analog DUT output currents come from the DUT power supply. If digital currents flow through the same return path as analog currents, the analog ground reference level (used for important analog measurements and digitizing) can be disturbed. Keep analog and digital currents separated as much as possible.

NOTES:

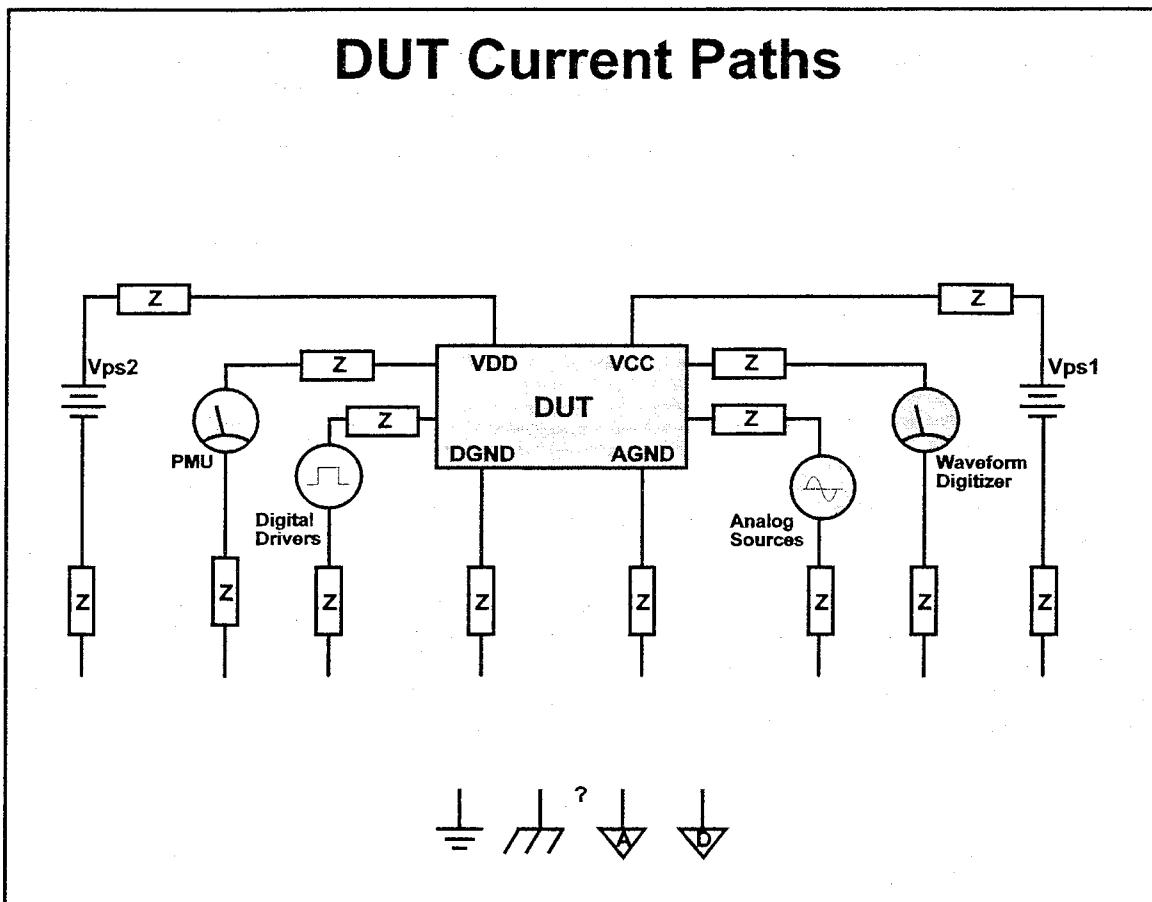


Figure 9.1—How many ways can you connect ground?

Figure 9.1 illustrates the many possible currents that must be returned to their various sources. ATE manufacturers approach ground current return in different ways. There may be more than one ground return path but normally not all return paths are separate. Often there is simply a “ground plane” available for use, and you must use the assumption that it has zero impedance.

There is an approach to grounding called the *star* ground which assumes that all available return paths are connected to a single common point. With several wires radiating from a single point, it looks like a star, thus the name. An excellent discussion of ADC ground layout and routing is given at the end of the chapter.²

NOTES:

The tester's earth ground connection

On multiple occasions, strange intermittent test failures have been fixed by making a new earth ground connection very close to the ATE system. One occasion solved a test system ground level problem in which 60Hz AC was coupled into one part of the test system but not another part. Another occasion solved a problem in which RF signals were being received by the system and not properly shunted to earth, thus allowing the RF to interfere with signals being digitized at the test head.

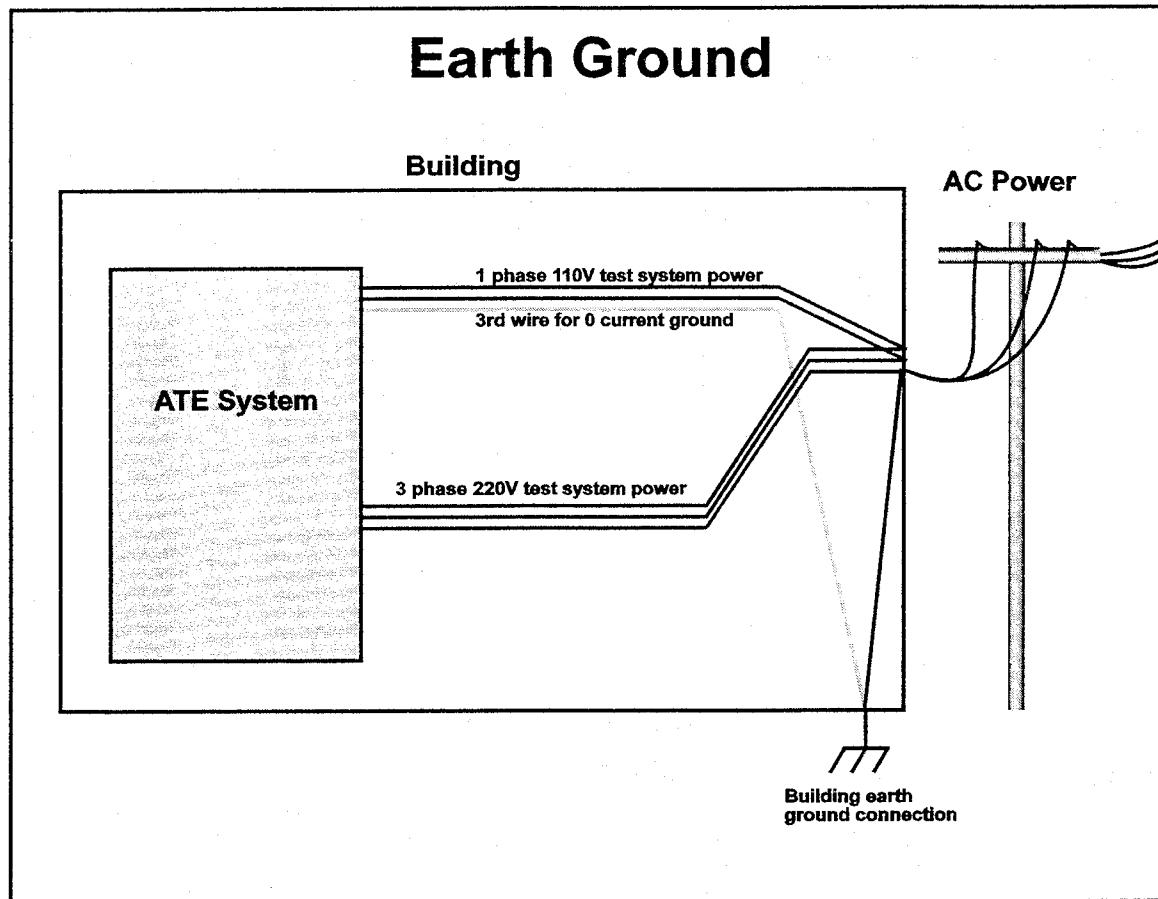


Figure 9.2

NOTES:



Notice in Figure 9.2 that a zero current ground return is available (in the US, at least) that truly reflects the voltage at the building's earth ground connection. However, the 220V 3 phase power lines are shown to not use this zero current ground. Thus any impedance in the neutral line of the ATE system will raise the neutral voltage at the test system as compared to the building's earth ground connection. If the ATE system uses that neutral as a reference node for anything, that reference "ground" will have a different voltage than the zero current "ground" connection. How different? Consider 30 amps of return current through 0.5 ohms—15 volts different! A local earth connection significantly reduces this problem (a copper, iron or steel pipe that goes from the tester through the floor and down to the underground "water table").

NOTES:

DUT power supply

As shown in Figure 9.1 on page 313, the DUT power supply has a finite output impedance and the signal path from the supply to the DUT has impedance. In many cases, a single power supply is required to deliver transient current for digital portions of the DUT while remaining solid and stable for analog portions of the DUT. The first and foremost requirement is that the DUT power supply be capable of delivering the total current required by the DUT. By using decoupling capacitors, the dynamic current requirements for the supply are lessened, although the supply must be able to replenish the charge in the decoupling capacitance before it "runs out of electrons."

Power Supply Decoupling

For CMOS digital signals, current is required only to charge capacitance when a logic circuit changes state. Thus current is needed by the DUT only during the short time for a logic transition. In general, we can assume that this will occur somewhere in the circuit on every clock edge. To qualify as a "digital" circuit, the elapsed time during clock edges must be a lot less than the time between the edges. Let's assume that a positive edge and negative edge use 20% of the total clock period. That leaves 80% of the period with little current drawn by the DUT.

By placing a "decoupling" capacitor at the DUT between the power supply pin and ground, the charge needed to supply those transient currents during 20% of the clock period can come from the capacitor. The power supply can replenish the current to the capacitor during the other 80% of the clock period when little current is required. This also decreases the high speed transient currents flowing through the signal path from the power supply to the DUT, thus "decoupling" that impedance from the DUT. The decoupling effect reduces transient voltage fluctuations at the DUT supply pin due to transient currents through the supply path impedance.

So what's a good value to use as a decoupling capacitor? Like most every other question in mixed signal testing, the answer is "it depends on the circuit and the system." The initial thought might be "the bigger, the better" except if the capacitor is too big it can slow the DUT VDD voltage too much. In other words, if the decoupling cap is too big, the DUT supply voltage will take too long to reach its specified value when the DUT is initially powered. Also, ATE power supply drivers may not be as robust as a benchtop power supply and may oscillate if loaded too heavily with capacitance.

Also, capacitors have parasitic characteristics that affect their performance. Some of these are *equivalent series resistance* (ESR), *dielectric absorption* (DA), *temperature coefficient of capacitance* (TC), *leakage current* and *dissipation factor*. Different characteristics are important in different situation, e.g. the important characteristics for a capacitor used in a switch mode power supply are different from those used for supply decoupling.

Rule of thumb for decoupling capacitors

As a general rule of thumb, a good starting point for decoupling a DUT is a $10\mu\text{F}$ tantalum cap in parallel with a $0.1\mu\text{F}$ ceramic cap. The tantalum cap provides lots of energy storage, while the ceramic cap can better deliver

NOTES:

current at high frequencies. Be careful when connecting the tantalum—it is polarized and can act like a forward biased diode if connected backwards.

Ceramic caps are available in widely different materials, e.g. X7R, Z5U, NPO. The primary distinction between the materials is temperature coefficient of capacitance. Also, some materials have a higher dielectric permittivity, meaning higher capacitance values are available in a smaller physical size component. Check the manufacturer's data or component retailer's product catalog for specifics on capacitor specifications.

Where to put the capacitor

Locating the decoupling capacitor(s) is a simple matter—put them as close to the DUT power and ground pins as you can. With a DUT having separate analog and digital ground pins, there must be a common point where the 2 grounds are connected. This point should also be as close to the DUT as possible, so connect the negative side of all decoupling caps there.

If the DUT also has separate power supplies, e.g. $\pm 5V$ or $+5V$ and $+15V$, decouple all supplies from the supply pin to ground with separate capacitors. If you are building test hardware for wafer probing, the decoupling capacitors should be on the probe card itself if at all possible. For handler hardware, the decoupling caps should be placed as close to the DUT site on the handler as possible. For a hand test site, place the caps directly underneath the DUT socket.

An excellent discussion of power supply by-passing can be found in *Electronic Design* magazine.³ It specifically discusses op amps but has general applicability.

NOTES:

Temperature Effects

Most data converters are temperature sensitive. That is, for a given code, a converter's input (ADC) or output (DAC) value will change as the DUT's changes temperature. The amount of change is denoted as temperature sensitivity or *drift*, usually in $\text{ppm}/^{\circ}\text{C}$, which is *parts per million per degree C*. Recall that *ppm* is similar to percent (parts per hundred), except based on one million rather than one hundred.

Question 9.2: A 16 bit DAC with 5.120V FSR has an offset voltage error measured at 0.01376V at 25°C and a temperature drift of offset of -135(ppm of FSR)/°C. What is the value of the offset voltage at 0°C and at 70°C?

Question 9.3: How many LSBs does the offset change over the 70° temperature change?

Question 9.4: How many LSBs does the offset change per 1° temperature change?

Offset, gain and/or linearity can change with temperature.⁴ In a production test situation, there are many factors that can affect the temperature of a DUT:

- ◊ Air temperature
- ◊ Air flow at the DUT site
- ◊ DUT socket temperature
- ◊ DUT package material (metal and ceramic absorb/dissipate heat faster than plastic)
- ◊ How long the DUT is powered (DUT self heating); related to DUT power usage
- ◊ Where the DUT is stored prior to testing

These factors must be closely monitored if repeatability problems exist.

NOTES:



DUT Reference Signal

ADCs and DACs require some sort of reference signal, usually a DC voltage. This signal must usually be "steady like a rock," not changing with time, temperature, power supply ripple, or reference output current. Real references are not quite this steady, however, and can be affected in one way or another by all of the above.

Internal Reference

Often a converter has an internally generated reference signal. This is the easiest case to test, as it is what it is. The only way to measure its effect is to measure full scale, which is directly dependent on the reference signal. If the reference signal is brought to an external pin, the reference voltage can be tested directly. It can also be distorted by using it as a reference for other circuitry on the DUT board or in the tester.

For example, if your test system has a reference input for its waveform digitizer, you may be able to drive the digitizer's full scale reference input with the DAC's reference out to make the WD match the DUT's full scale. The digitizer is a dynamic device, however, with a sample clock, an A/D convert signal, etc. The dynamic demands of the digitizer may cause glitches, ripple, etc. on the DUT reference signal, distorting the reference voltage and the DAC output. It is best to use a high speed, accurate buffer between a DUT's reference output and any external circuitry.

Also, internal references are not designed to supply current. If a DUT's internal reference is loaded too heavily, its output value can change or go out of specification limits.

External Reference

If the DAC under test does not have its own reference, you must supply one. Be sure to use a high accuracy device with low drift. Connect its ground directly to the DUT ground with little or no wire or trace lengths between them. Look at the reference output with a high speed oscilloscope to make sure that it is stable at all frequencies. Look at it while under a test loop to make sure there is not excessive digital noise being coupled into the reference output signal. If the DUT socket is designed for temperature test, place the reference device in a place where it will not be affected by DUT temperature. If it requires accuracy trim, use 0.1% low TC (temperature coefficient) metal film resistors.

NOTES:

Averaging and repeatability

With low level measurements, noise can have a significant effect, both device noise and test system noise. Particularly, an instantaneous measurement (such as one made with a digitizer) can be affected by a noise peak. Random noise will average to zero, so averaging a number of measurements can remove uncertainty due to random noise. This is especially helpful when an error signal is created by using an amplified difference between a "perfect" reference signal and the signal being measured. The trade-off for averaging is test time.

Digital noise may not average to zero; that is, noise due to digital transients that are coupled into the signal being measured may not be helped by averaging. This is why a clean DUT ground and careful attention to signal routing and current flow is so important!

The level of repeatability is usually determined during the device characterization stage. Repeatability requirements are set by the device, the parameter and company or department policy. Averaging can be used in both static and dynamic tests. Averaging for static tests means testing the same point on a transfer curve multiple times. Averaging for dynamic tests means taking the same sample points over multiple test cycles.

NOTES:



Trouble-shooting: look for obvious problems first

You've created a test program, test hardware and have independently correlated it to NIST calibrated sources. Now the test is in production and devices will not pass. To avoid wasting time, when a problem occurs look for the obvious first. For example, if a device that has worked in the past now fails, check these items:

- ❖ Is the correct test program loaded?
- ❖ Is the correct DUT temperature setting entered?
- ❖ Is the correct test hardware mounted?
- ❖ Is the test hardware properly seated?
- ❖ Is the DUT socket closing properly on the DUT pins?
- ❖ Is there a bent or damaged DUT pin?
- ❖ Is the handler configured for the correct DUT package?
- ❖ Is the probe card damaged?
- ❖ Is the prober microscope light on?
- ❖ Is the prober stepping the correct distance for the die being tested?
- ❖ Has the wafer been properly etched so the bond pads are exposed?

After verifying that these things are not the source of the problem(s), set up a device in a test loop or pause at a specific program location and check, with a meter or oscilloscope, these items in this order:

- ❖ Power supply voltages
- ❖ All analog signal sources
- ❖ Signals at digital input pins
- ❖ Signals at DUT analog output pins
- ❖ Signals at DUT digital output pins

Naturally, you can save time by looking at a device datalog to get information about where to start. For example, if a DAC fails a digital input leakage test and a distortion test, chances are good that a digital input pin is causing both problems.

NOTES:

Unexpected things can affect measurement results

Sometimes things go wrong with a device test and none of the usual suspects can be found as the problem. There are other, less likely suspects that you may not have even considered. Here we consider them, with some "war stories" as support.

An ungrounded DUT package

A DAC once failed a digital feed-through test; the digital signals on the DAC inputs were coupling somehow to the DAC output. The DAC internal layout was done very carefully—no digital lines were near any high impedance analog nodes. The DUT board was designed carefully, with a good star ground and careful decoupling. Eventually, the problem was traced to the DUT package. The DAC was packaged in a ceramic DIP with a metal lid. The digital signals were capacitively coupling to the lid and back into the DAC's summing junction. A new device package had to be made that connected the lid to the DUT's internal ground, solving the feed-through problem. Capacitive coupling can also affect signals in other ways; be aware that any 2 conductors separated by an insulator makes a capacitor.

Another problem that can occur with packages is that of getting signals from the inside to the outside of the package. Gold bond wires in particular have a high thermal coefficient of resistance. If there are problems with DUT zero shift when testing a device over temperature, perhaps the package leads or the bond wires are causing a ground resistance change over temperature.

External noise from RF or magnetic sources

With cellular phones, pagers, PDAs and other fancy gadgets proliferating, there is more high frequency content in the air than ever. With so many carrier frequencies being broadcast by microwave towers and satellites, virtually any metal object becomes an antenna. Use a good broad frequency spectrum analyzer to characterize your test floor and its DUT sites to make sure they are not picking up radio or microwave signals and causing measurement and digitizing errors. This is especially important for test situations which use wide band amplifiers, either in the DUT or in the test hardware.

Light

A photocell consists of a *p-n* junction. So does every known silicon IC (well, many *p-n* junctions). If *any* light falls on a silicon circuit, the circuit behaves differently than it would in a dark package. Light may not have much effect on digital circuits, but analog and mixed signal circuits can have significant current and voltage differences in different light conditions. When doing wafer or open package testing, make sure that NO LIGHT gets to the DUT. If you characterize prototypes with no package lid and the lights are on, your characterization data is probably flawed.

NOTES:



Humidity

Water conducts current. If humidity is not controlled, low level measurements of mixed signal and analog circuits, not to mention digital input leakage tests, can be wrong by large amounts. A load board stored in your office for a week may cause test failures after being returned to the production area because it has absorbed moisture. *Keep all test circuitry and PC boards in a controlled humidity environment!*

NOTES:

Answers to chapter questions

Answer 9.1: Longer test time. On the bright side, most ATE systems have a built in array processor routine which will average a set of sample arrays very quickly. There is no avoiding the time required for multiple ADC conversions over multiple UTPs to gather the data, however.

Answer 9.2: Temperature drift in volts: $5.120 \times -135 \times 10^{-6} = -0.6912 \text{mV}/\text{°C}$

From 25 to 70 degrees is a positive 45° change, meaning a negative voltage drift, so offset change $-0.6912 \text{mV}/\text{°C} \times 45\text{°C} = -0.031104 \text{V}$

Offset at 70°C = offset at 25°C + offset change = $0.01376 - 0.031104 = -17.344 \text{mV}$

From 25 to 0 degrees is a negative 25° change, meaning a positive voltage drift, so offset change $0.6912 \text{mV}/\text{°C} \times 25\text{°C} = 0.01728 \text{V}$

Offset at 0°C = offset at 25°C + offset change = $0.01376 + 0.01728 = +31.04 \text{mV}$

Answer 9.3: Voltage change over 70°C = $31.04 - (-17.344) = 48.384 \text{mV}$

Change in LSBs = $48.384 \text{mV} / (5.120 / 65536) \text{V per LSB} = 619.3 \text{ LSBs (!)}$

Answer 9.4: Voltage change over 1°C = $619.3 \text{ LSBs} / 70\text{°C} = 8.8 \text{ LSB}/\text{°C}$. Thus if a device changes temperature by 1°C during measurements for a linearity test, it will fail.

NOTES:

References

1. Paul Brokaw, *An I.C. Amplifier Users' Guide To Decoupling, Grounding, And Making Things Go Right For A Change*, Analog Devices Application Note AN-202, Timeless. Available as of this writing at http://www.analog.com/techsupt/application_notes/application_notes.html
2. William C. Rempfer, Get All the Fast ADC Bits You Pay For, *Electronic Design Analog Applications Issue*, June 24, 1996, pp 9-25.
3. Gerald Graeme and Bonnie Baker, Design Equations Help Optimize Supply Bypassing for Op Amps, *Electronic Design Analog Applications Issue*, June 24, 1996, pp 9-25.
4. Donald S. Bruck, *Data Conversion Handbook*, First Edition, Hybrid Systems Corp. 1974

NOTES:



General Test Issues

NOTES:

Index

A

- AC measurements 16
- accuracy, ADC 95
- accuracy, DAC 62
- acquisition time 277
- ADC architectures 114
- ADC dynamic test, steps 267
- ADC input, anti aliasing 271
- ADC input, clipping 268
- ADC input, filtering 269
- ADC input, purity requirements 270
- ADC noise 310
- ADC, test problems 102
- ADC, zero/full scale adjust 268
- Alexander Graham Bell 129
- Aliasing 172
- aliasing, effect of 173
- aliasing, preventing 174
- amplifier noise 310
- analog ATE 25
- analog building blocks 24
- angular frequency and theta 136
- angular velocity 136
- aperture jitter 96
- aperture time 95, 273
- array processor 32
- averaging and repeatability 320

B

- bandpass sampling 165
- bandwidth, filter 52
- beat frequency sampling 287
- bin 185
- bipolar 60
- bit 3
- Boolean logic 7

C

- C. E. Shannon 165
- capacitor, decoupling 316
- capture memory 34, 37
- capture RAM 37
- center frequency 52
- center of code 102
- center of code, finding 115
- chip select 64
- clock 64
- CMRR 47
- codes to test, ADC 114
- coherency 184
- coherent sampling 184
- common log 125
- common mode rejection ratio 47
- comparator, specifications 51
- Complex numbers 155
- complex plane 155
- conversion time 95
- cosine 138
- creating ADC sine wave input 268
- cross point matrix 26
- current output DAC 84
- current path 313

D

- DAC dynamic test steps 225
- DAC output samples 226
- DAC test, system configuration 64
- DAC, output filtering 235
- data storage memory 34, 36
- Data Valid 100
- dB 129
- dB ratio table 130
- debugging 321
- decibel 129
- decoupling 316
- DFT 169
- differential nonlinearity, ADC 94
- differential nonlinearity, DAC 62
- digital semiconductor testing 45
- digital test systems 9
- discrete Fourier Transform 169
- distortion, harmonic 48
- distortion, $\sin(x) / x$ 179
- DNL measurement, DAC 74, 75
- DNL, ADC 94
- DNL, DAC 62, 82
- DNL, equation for static, ADC 116
- DSP 32, 37
- DSP advantages 29
- DUT LSB size, ADC 102
- DUT package 322
- DUT power supply 316
- Dynamic 45
- Dynamic IDD 45
- dynamic impedance 277
- dynamic range 262
- dynamic specifications, DAC 211

E

- earth ground 314
- Effective Number of Bits 118
- effective number of bits, derived 297
- end-point linearity, DAC 83
- ENOB 118
- ENOB derived 297
- envelope sampling 287
- equivalent power 141
- exponent 5
- External Reference 319

F

- family board 28
- Fast Fourier Transform 169
- FFT 169
- FFT, using 170
- filter bandwidth 52
- filter settling time 52
- floating point 5
- fmax 45
- Fourier frequency 185
- frequency bin 185
- frequency domain 143
- frequency resolution 185
- frequency spectra 143
- Fres, defined 185
- Fs 185
- FSR equation, ADC 103
- FSR, ADC 92
- FSR, DAC 60, 220
- Ft 185
- full scale input value, ADC 104
- full scale range equation, ADC 103
- full scale range, ADC 92
- full scale range, DAC 60, 220
- functional testing 11
- fundamental correction 248
- fundamental, bin number 185
- fundamental, removing 242

G

- gain bandwidth 47
- gain correction 247
- gain error, ADC 93
- gain error, ADC, calculating 108
- gain error, DAC 61
- gain error, equation, DAC 72
- GBW 47
- go-no go 11
- grounding 312

H

- half sine Fourier series 147
- harmonic distortion 48
- Harry Nyquist 165
- high accuracy reference source 80
- highest frequency of interest 165
- histogram distribution 111
- Humidity 323
- hysteresis, ADC 106



Index

I

- IDD 45
IFFT 197
IFFT, how to use 198
IFFT, limitations 200
IFFT, when to use 198
IIB 46
IIH 45
IIL 45
IIO 47
IM 218
incomplete sample set 176
INL calculation, ADC 117
INL test, DAC 83
INL, ADC 95
INL, DAC 62
Input bias current 46
input offset current 47
input offset voltage 46
integer 5
integral nonlinearity 95
integral nonlinearity, DAC 62
intermodulation dist'n, ADC 262
intermodulation dist'n, DAC 217
intermodulation distortion 218
Internal Reference 319
inverse FFT 197
IOH 45
IOL 45
IOS 45
IOZ 45

J

- Jitter Error 183

L

- leakage 175
level shifting, DAC output 240
Light 322
log base 2 127
Log base conversion 128
logarithm base 125
Logarithms 125
logic gates 7
low pass filter, DAC 236
LSB, defined-, DAC 62
LSB, defined-ADC 93

M

- M 185
magnitude calculation 248
mantissa 5
maximum conversion rate 63, 220
missing codes 95
mixed signal circuit types 31
mixed signal test system 33
monotonic 62
monotonicity 62
mutually prime 187

N

- N 185
narrow codes 111
natural log 126
no missing codes 95
noise, calculating 48
noise, defined 47
noise, RF 322
notch filter 242
Nyquist's theorem 165

O

- obvious problems 321
offset error equation, DAC 69
offset error, ADC 92
offset error, ADC, calculating 107
offset error, DAC 60
Ohm's law 15
op amp 46

P

- package 322
parallel 3
parametric testing 14
parts per million 69
percent of full scale 69
period 131
Periodic motion 131
Periodicity 167
phase relationship 138
pin drivers 14
pin electronics 14
PMU 15
polar / rectangular conversion 158
power ratio 129
power supply decoupling 316
power supply rejection ratio 47
PPM 69
PSRR 47

Q

- Q, filter 52
quality factor, filter 52
Quantization Error 181

R

- R/2R DAC 75
rack and stack 25
reference signal 319
Replication 172
resolution, DAC 60
response time 51
RF noise 322
Richter Scale 125
RMS 140
root mean square 140

S

- sample set 167
samples, generating 197
sampling with ADC 279
sampling, coherency equation 185
sampling, coherent 184
sampling, F_s 185
sampling, F_t 185
sampling, M 185
sampling, N 185
scaling, DAC output 240
serial 3
settling time 49
settling time errors 311
settling time, DAC 63, 220
settling time, filter 52
SFDR, ADC 262
Shannon's theorem 165
signal to noise & dist'n, ADC 262
signal to noise & dist'n, DAC 214
signal to noise and distortion 49
signal to noise ratio 49
signal to noise ratio, ADC 262
signal to noise ratio, DAC 220
 $\sin(x) / x$ distortion 179
SINAD 49
SINAD, calculating 214
SINAD, DAC 214, 262
sine histogram 293
sine wave from pendulum 133
sine wave from rotating vector 134
sine wave points, algorithm 228
sine wave, generating DAC codes 226
sine wave, points on 226
sinusoid 132
slew rate 49
Slew Rate Error 182
SNDR, calculating 214
SNDR, DAC 214, 262
SNR 49
SNR, ADC 262
SNR, calculating 215
SNR, DAC 215, 217, 262
spectrum analyzer 168
spurious free dynamic range 262
square wave Fourier series 147
SR 49
Start Convert 100
start convert 97
state machine 7
straight line, equation 83
sub-Nyquist sampling 165
summing junction 68
superposition error 78
synchronization 251
system measurement req'ts 311
system noise 309

T

- tdis 45
- temperature effects 318
- ten 45
- test problems, ADC 102
- th 45
- THD 48
- THD, ADC 262
- THD, calculating 216
- THD, DAC 216
- time window functions 176
- time windows, reducing leakage 176
- total harmonic dist'n, ADC 262
- total harmonic dist'n, DAC 216
- total harmonic distortion 48
- tpd 45
- track and hold 275
- transition noise 96
- transition noise, ADC 106
- transition points 102
- transitions, find with step DAC 104
- triangle wave Fourier series 147
- Truncation error 181
- ts 49
- tsu 45

U

- undersampling 284
 - unipolar 60
 - unit test period 185
 - UTP, defined 185
- V**
- vector memory 11
 - vector sequencing 17
 - VIH 45
 - VIL 45
 - VIO 46
 - VOH 45
 - VOL 45

W

- waveform digitizer 34
- waveform generator 35
- WD, example spec 240
- wide codes 111
- window, Blackman 177
- window, Blackman-Harris 177
- window, Hamming 177
- window, Hann 177
- window, rectangular 177
- window, triangular 177
- windowing functions 177
- write 64

Z

- zero scale input, ADC 103
- zero scale, DAC 67

Topics Index